A FAST ALGORITHM FOR SOLUTION
OF TRANSPORTATION PROBLEMS

Ramesh Gupta

TECHNICAL REPORT
in
The Faculty
of
Engineering

Presented in Partial Fulfillment of the Requirements for
the Degree of Master of Engineering at
Sir George Williams University
Montreal, Canada

September, 1972

© Ramesh Gupta 1973
TABLE OF CONTENTS

ABSTRACT..........................................................................................vi
ACKNOWLEDGEMENTS...........................................................................vii
LIST OF SYMBOLS..................................................................................viii
INTRODUCTION.......................................................................................1

CHAPTER 1

BACKGROUND.......................................................................................2

CHAPTER 2

FORMULATION OF TRANSPORTATION PROBLEM........................................5
The Transportation Problem Table.........................................................8

CHAPTER 3

LOOPS AND TREES............................................................................12
Element or cell......................................................................................12
Directed Path Joining Two Cells............................................................12
Directed Loop......................................................................................15
Tree....................................................................................................15
Basis....................................................................................................15
CHAPTER 9
INEQUALITIES IN THE CONSTRAINTS OF A TRANSPORTATION PROBLEM.......... 73
Availability Greater than Demand................................................. 75

CHAPTER 10
OPERATING EXPERIENCE............................................................. 78
Method 1: Best Position in Cost Matrix........................................ 79
Method 2: First Which Would Reduce Total Cost............................ 80
Method 3: Best in Row of Cost Matrix.......................................... 80
Comparison of Methods.............................................................. 80
Comparison of Solution Time Ratios............................................ 84

CONCLUSION.............................................................................. 85

BIBLIOGRAPHY........................................................................... 86

APPENDIX
A. Flow Charts............................................................................ 89
B. Program Listings...................................................................... 103
A FAST ALGORITHM FOR SOLUTION OF
TRANSPORTATION PROBLEMS

ABSTRACT

The transportation problem has been formulated by various investigators and solved to various degrees. The systematic method of solution was first given by Dantzig. In general, the computational procedures are adaptation of the simplex method. However, almost all of the techniques either take too long to be solved by a digital computer or are not readily adaptable for use on digital computers.

The northwest corner rule has been presented for solving the transportation problem. The essentials of the stepping stone method are then reviewed. This technique does not consider costs for determining the initial basic feasible solution. Other techniques which make some use of costs are also described.

A modified technique for solving transportation problem by digital computer is then presented along with the unique features of the method which give its high speed in solving problems. The detailed logic of the method is also explained. This method was implemented and was found to reduce the solution time by 2.6 times as compared to the well known matrix minima method of solution.
ACKNOWLEDGEMENTS

The author wishes to express his appreciation to his supervisor, Prof. G. Martin for his encouragement and guidance throughout all stages of this work.

The author extends special thanks to Sir George Williams University for providing computer time which made this entire work possible.
LIST OF SYMBOLS

\[ a \] - Basis table entry number of the search table.
\[ b \] - Basis table entry number of the branch point table.
\[ i \] - Row of basis table.
\[ j \] - Column of basis table.
\[ m \] - Number of sources (plants).
\[ n \] - Number of destinations (customers).
\[ t \] - Entry number of element.
\[ Z \] - Objective function.
\[ \Delta \] - Perturbation.
\[ A_a \] - Identifies \( F_a \) encountered during search
\[ A_a = 0 \] - corner point
\[ = 1 \] - branch point
\[ = 2 \] - branch point from which a new path was searched.

\[ B_b \] - Indicates whether branch point was found during
\[ B_b = 0 \] - row search
\[ = 1 \] - column search.

\[ C_{ij} \] - Unit cost from source \( i \) to destination \( j \).
\[ \tilde{C}_{ij} \] - Indirect cost matrix.
\[ \Delta C \] - Incremental cost.
\[ D_j \] - Demand of commodity by customer \( j \).
\[ F_a \] - Element number of branch or corner point encountered during search.
$G_b$ - Element number of branch point encountered during search.

$I_t$ - Row of basis element $t$.

$J_t$ - Column of basis element $t$.

$K_t$ - Number of an element in the same row of shipment matrix as element $t$.

$L_t$ - Number of an element in the same column as an element $t$.

$S_i$ - Supply of commodity at source $i$.

$U_i$ - Dual variable for row $i$.

$V_j$ - Dual variable for column $j$.

$X_{ij}$ - Commodity shipped from source $i$ to customer $j$. 
INTRODUCTION

The purpose of this report is to present and implement a method of solution to a class of problems known as the transportation problems.

The report presents a brief history and definitions of the common terms used. It then describes in detail the stepping stone method which was implemented by the author. This new logic was found to provide a speed improvement over usual methods. Program listing and flow charts are provided in the appendixes.
CHAPTER 1

BACKGROUND

One of the earliest and most fruitful applications of linear programming techniques has been the formulation and solution of the transportation problems as a linear-programming problem.

L.V. Kantorovich showed that a class of problems closely related to the classical transportation case has a remarkable variety of applications concerned typically with the allotment of tasks to machines whose costs and rates of production vary by task and machine type \textsuperscript{(5)}*. He gave a useful but incomplete algorithm for solving such problems. In 1942, he wrote a mathematical paper concerned with a continuous version of the transportation problem, and in 1948, he authored an applicational study, jointly with Gavurin, on the capacitated transportation problem.

The now standard form of the problem was first formulated, along with a constructive solution, by Frank L. Hitchcock. His paper, "The Distribution of a Product from Several Sources to Numerous Localities" \textsuperscript{(18)},

\* Represents bibliography reference number.
sketched out the partial theory of a technique foreshadowing the simplex method; it did not exploit special properties of a transportation problem except in finding starting solutions. This paper also failed to attract much attention.

Still another investigator, T.C. Koopmans, as a member of the Combined Shipping Board during World War II, became concerned with using solutions of the transportation problem to help reduce overall shipping times, for the shortage of cargo ships constituted a critical bottleneck\(^{(10)}\).

In 1947, Koopmans began to spearhead research on the potentialities of linear programs for the study of problems in economics. His historic paper, "Optimum Utilization of the Transportation System\(^{(20)}\), was based on his wartime experience. Because of this and the work done earlier by Hitchcock, the classical case is often referred to as the Hitchcock Koopmans Transportation Problem.

Another, whose work anticipated the recent era of development in linear programming was E. Egervary, a mathematician. His 1931 paper considered the problem of finding a permutation of ones in a matrix composed of zero and one elements\(^{(12)}\). Based on this investigation, Kuhn developed an efficient algorithmic method for solving assignment problems\(^{(21)}\). Kuhn's approach, in its turn, underlies the Ford-Fulkerson Method for solution of the classical transportation problem\(^{(11)}\).

The linear-programming formulation and the associated systematic method of solution were first given by Dantzig\(^{(5)}\). The computational
procedure is an adaptation of the simplex method applied to the system of equations of the associated linear-programming problem.

This report describes the implementation of a new digital computer technique for solving the classical transportation problem by the stepping stone method. This technique offers considerable advantage in speed over methods currently in use\(^{(6,15)}\). First, the essentials of the stepping stone method are reviewed. Then, the unique features of the method which give it high speed in solving problems with it, are presented. The detailed logic of the method is also given.
CHAPTER 2

FORMULATION OF TRANSPORTATION PROBLEM

The general transportation problem may be formulated as follows:

A Company operates \( m \) plants ("origins") producing a commodity, the \( i \)th of which can supply \( S_i \) units. The company sells its production to \( n \) customers ("destinations"), the \( j \)th of which demands \( D_j \) units of the commodity. The cost of manufacturing and transporting a unit of the commodity from plant \( i \) to customers \( j \) is \( C_{ij} \). It is desired to find the number of units \( X_{ij} \) that should be shipped from each plant to each customer so that the total cost of the operation is a minimum.

To develop the constraints of the problem, set up Table 2.1. The amount shipped from source \( i \) to destination \( j \) is \( X_{ij} \), the total shipped from source \( i \) is \( S_i \geq 0 \) and the total received by destination \( j \) is \( D_j \geq 0 \).

Imposing temporarily the restriction that the total amount shipped is equal to the total amount received, that is:

\[
\sum_i S_i = \sum_j D_j = A \tag{2.1}
\]

The total cost of shipping \( X_{ij} \) units is \( (C_{ij} \cdot X_{ij}) \). Since a
### TABLE 2.1: Transportation Problem Table

<table>
<thead>
<tr>
<th>DESTINATIONS</th>
<th>SOURCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>(j)</td>
<td>(1)</td>
</tr>
<tr>
<td>(1)</td>
<td>x_{11}</td>
</tr>
<tr>
<td>(2)</td>
<td>x_{21}</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>(1)</td>
<td>x_{11}</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>(m)</td>
<td>x_{m1}</td>
</tr>
<tr>
<td></td>
<td>D_1</td>
</tr>
</tbody>
</table>
negative shipment has no valid interpretation for the problem, each
\( x_{ij} \geq 0 \).

Table 2.1 gives the mathematical definition of the problem:

Find values for the variables \( x_{ij} \) which minimize the total cost:

\[
Z = \sum_{ij} c_{ij} x_{ij}
\]

Subject to the constraints:

\[
\sum_j x_{ij} = s_i \quad i = 1, 2 \ldots, m
\]

\[
\sum_i x_{ij} = d_j \quad j = 1, 2 \ldots, n
\]

and

\[
x_{ij} \geq 0
\]

Equation 2.3 represents the row sums of Table 2.1 and equation 2.4
the column sums.

In order for equation 2.3 and 2.4 to be consistent, the sum of
equation 2.3 must be equal to the sum of equation 2.4.

That is,

\[
\sum_i \sum_j x_{ij} = \sum_j \sum_i x_{ij} = \sum_i s_i = \sum_j d_j = A
\]

System of equations 2.2 to 2.5 is a linear programming problem
with \( m + n \) equations in \( mn \) variables.
For \( m = 3 \) and \( n = 5 \), writing the equations corresponding to equations 2.3 and 2.4, gives 8 equations (that is \( m + n \)) in 15 (that is \( mn \)) unknowns as shown in Table 2.2.

**THE TRANSPORTATION PROBLEM TABLE:**

Consider the Table 2.3. In cell \((i,j)\) enter \( c_{ij} \) and \( x_{ij} \). If the \( x_{ij} \) entered in the table represents a feasible solution, it must be true that addition of the \( x_{ij} \) in row \( i \) yields \( s_i \), for \( i = 1, \ldots, m \). Similarly, addition of the \( x_{ij} \) in column \( j \) yields \( d_j \), for \( j = 1, \ldots, n \). Hence, all the constraints are conveniently represented, and it is easy to check whether any set of \( x_{ij} \), is a feasible solution by simply summing the rows and columns.

In the last column enter the origin availabilities and in the last row the destination requirements. It is also convenient to use \( d_j \) as a heading for column \( j \) to indicate that this column pertains to destination \( j \). Similarly \( s_i \), is placed at the beginning of row \( i \) to indicate that this row pertains to origin \( i \).

In the lower right-hand cell of Table 2.3, enter the total amount to be shipped, i.e., \( \sum s_i = \sum d_j \).
\[ x_{11} + x_{12} + x_{13} + x_{14} + x_{15} = S_1 \]
\[ x_{21} + x_{22} + x_{23} + x_{24} + x_{25} = S_2 \]
\[ x_{31} + x_{32} + x_{33} + x_{34} + x_{35} = S_3 \]
\[ x_{11} = D_1 \]
\[ x_{21} = D_2 \]
\[ x_{31} = D_3 \]
\[ x_{11} = D_4 \]
\[ x_{21} = D_5 \]

**TABLE 2.2:** Equations 2.3 and 2.4 for \( m = 3, n = 5 \)
<table>
<thead>
<tr>
<th></th>
<th>$D_1$</th>
<th>$D_2$</th>
<th>...</th>
<th>$D_J$</th>
<th>...</th>
<th>$D_n$</th>
<th>$S_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>$c_{11}$</td>
<td>$x_{11}$</td>
<td>...</td>
<td>$c_{1j}$</td>
<td>...</td>
<td>$c_{1n}$</td>
<td>$x_{1n}$</td>
</tr>
<tr>
<td></td>
<td>$c_{21}$</td>
<td>$x_{21}$</td>
<td>...</td>
<td>$c_{2j}$</td>
<td>...</td>
<td>$c_{2n}$</td>
<td>$x_{2n}$</td>
</tr>
<tr>
<td></td>
<td>$x_{31}$</td>
<td>...</td>
<td>...</td>
<td>$x_{3j}$</td>
<td>...</td>
<td>$x_{3n}$</td>
<td>$S_2$</td>
</tr>
<tr>
<td>$S_n$</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>$S_m$</td>
</tr>
<tr>
<td>$D_j$</td>
<td>$x_{m1}$</td>
<td>$x_{m2}$</td>
<td>...</td>
<td>$x_{mj}$</td>
<td>...</td>
<td>$x_{mn}$</td>
<td>$S_j = D_j$</td>
</tr>
</tbody>
</table>

**TABLE 2.3:** Table for Transportation Problem (Including Costs)
When dealing with basic solutions, no more than $m + n - 1$ of the $X_{ij}$ in Table 2.3 will be positive. For basic feasible solutions, no more than $m + n - 1$ of the $X_{ij}$ will ever be $> 0$. Only the values of the basic variables will be entered in the table i.e., not fill in the zeros for the nonbasic variables. However, zero values of the basic variables will be written in.
CHAPTER 3

LOOPS AND TREES

In order to discuss the transportation problem in more detail, it is helpful to define certain terms:

ELEMENT or CELL:

An element is a position in the shipment matrix. The value of an element is the value of $X_{ij}$ for the position of the element in the shipment matrix. Elements of particular interest may be designated by their column, or may be arbitrarily numbered.

DIRECTED PATH JOINING TWO CELLS:

A directed path from the cell $(i,j)$ to the cell $(v,w)$ in Table 2.3 is defined to be an ordered set of cells $(i,j), (i,k), (q,k), (q,r), ..., (v,w)$ or $(i,j), (s,j), (s,t), ..., (v,w)$ such that any two adjacent cells in the ordered sets lie alternately in the same row, then in the same column, while any three adjacent cells do not lie in the same row or same column. Furthermore, each cell (except the last) must appear only once in the ordered sets. The cell $(i,j)$ is called the initial cell of the path, and $(v,w)$ is called the terminal cell.
Fig. 3.1  DIRECTED PATH

Fig. 3.2  DIRECTED LOOP
Fig. 3.3  EXAMPLE OF A BASIS

Fig. 3.4  LOOP INCLUDING NEW ELEMENT X
It is convenient to be able to illustrate graphically a path connecting two cells in Table 2.3 (shown in Fig. 3.1). To do this simply join by line segments the ordered set of cells which form the path. The direction is indicated by an arrowhead on the line. These line segments will be called branches.

**DIRECTED LOOP:**

A directed loop is a directed path such that the first cell in the ordered set is the same as the last cell, (that is, a path starting and terminating with the same cell). A typical directed loop is shown in Fig. 3.2.

**TREE:**

A tree is a connected set of cells without loops.

**BASIS:**

A basis is a tree in an m by n shipment matrix containing exactly \( m + n - 1 \) elements. A basis has the following properties:

a) There is at least one element of the basis in each row and each column of the shipment matrix.

b) Hence, if a new element is added to the tree, a unique loop is formed including that element.

An example of a basis and the loop formed by adding a new element is shown in Fig. 3.3 and 3.4.
CHAPTER 4

THE DUAL VARIABLES - CHANGE OF BASIS

In order to understand the transportation problem it is convenient to define an auxiliary variable $U_i$ associated with each row and a $V_j$ associated with each column of the cost matrix. These are the dual variables of linear programming theory. Their values are chosen so that

$$U_i + V_j = C_{ij} \quad (4.1)$$

for those combinations of $i$ and $j$ which correspond to elements of the basis.

Let $C_{ir}, C_{qr}, C_{qt}, \ldots, C_{ws}, C_{wj}$ be the $m + n - 1$ prices corresponding to the variables in any basic feasible solution to a transportation problem with $m$ origins and $n$ destinations. Now suppose that:

$$U_i + V_r = C_{ir},$$
$$U_q + V_r = C_{qr},$$
$$U_q + V_t = C_{qt},$$
$$\vdots \quad \vdots \quad \vdots \quad \vdots \quad (4.2)$$
$$U_w + V_s = C_{ws},$$
$$U_w + V_j = C_{wj}.$$

Since there are $m + n - 1$ elements in the basis and the number of $U$'s and $V$'s is $m + n$ resulting in one more unknown than number
### TABLE 4.1 (a) Shipment Matrix

<table>
<thead>
<tr>
<th></th>
<th>$D_1$</th>
<th>$D_2$</th>
<th>$D_3$</th>
<th>$D_4$</th>
<th>$D_5$</th>
<th>$D_6$</th>
<th>$S_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>3</td>
<td></td>
<td></td>
<td>5</td>
<td>5</td>
<td></td>
<td>13</td>
</tr>
<tr>
<td>$S_2$</td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>$S_3$</td>
<td></td>
<td></td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>$S_4$</td>
<td>3</td>
<td>2</td>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>$D_j$</td>
<td>3</td>
<td>7</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>36</td>
</tr>
</tbody>
</table>

### TABLE 4.1 (b) Cost Matrix

<table>
<thead>
<tr>
<th></th>
<th>$U_1$</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_1$</td>
<td>8</td>
<td>5</td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>$V_2$</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>$V_3$</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$V_4$</td>
<td>5</td>
<td>3</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>$V_5$</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$U_1$</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
of equations, any one of the variables can be set arbitrarily.

Suppose now that, an element is to be added in cell (3,5) of the shipment matrix in Table 4.1. The circled elements in Table 4.2(a) constitute a basic loop including the new element. If the new element is to have the positive value ΔX, the changes indicated in Table 4.2(b) must be made in the elements of the loop so that the supply and demand requirements remain satisfied. The change in total cost made by bringing in this new element including the effect of the changes in the loop elements is

\[ \Delta X (C_{35} - C_{15} + C_{12} - C_{22} + C_{23} - C_{33}) = -\Delta X \Delta C \]  

(4.3)

Using the definition of the U's and V's, this becomes

\[ -\Delta X \Delta C = \Delta X \left[ C_{35} - \left( U_1 + V_5 \right) + (U_1 + V_2) \right. \\
\left. - (U_2 + V_2) + (U_2 + V_3) - (U_3 + V_3) \right] \\
= \Delta X \left( C_{35} - U_3 - V_5 \right) 

(4.4)

Thus, if the quantity \( \Delta C = U_i + V_j - C_{ij} \) is negative or zero, no gain can be made by introducing an element in cell \( (i,j) \). However, if this quantity is positive, it will pay to make \( \Delta X \) as large as possible. For the present case \( \Delta C = 2 \) and a reduction of total cost may be obtained.

The value of \( \Delta X \) is limited by the element in cell (2,2) which becomes zero when \( \Delta X = 4 \). The new form of the shipment matrix is as in Table 4.3(a). Note that the non-zero elements again form with a tree containing exactly \( m + n - 1 \) elements which is therefore a new basis. Since the row and column sums of the shipment matrix are the same as before, the values of the basic elements give a new feasible solution.
\[
\begin{array}{cccccc|c}
D_1 & D_2 & D_3 & D_4 & D_5 & D_6 & S_1 \\
\hline
S_1 & 3 & & 5 & 5 & & 13 \\
S_2 & & 4 & 1 & & & 5 \\
S_3 & & & 7 & x & & 7 \\
S_4 & 3 & 2 & & 6 & & 11 \\
D_1 & 3 & 7 & 10 & 5 & 5 & 6 & 36 \\
\end{array}
\]

X - New Element.
○ - Element in Loop Including New Element.

TABLE 4.2 (a): Basic Loop Including New Element.

\[
\begin{array}{cccccc|c}
D_1 & D_2 & D_3 & D_4 & D_5 & D_6 & S_1 \\
\hline
S_1 & 3+\Delta X & 5 & 5-\Delta X & & 13 \\
S_2 & 4-\Delta X & 1+\Delta X & & & 5 \\
S_3 & & 7-\Delta X & \Delta X & & 7 \\
S_4 & 3 & 2 & & 6 & 11 \\
D_1 & 3 & 7 & 10 & 5 & 5 & 6 & 36 \\
\end{array}
\]

TABLE 4.2 (b): Changes in Elements of Loop.
<table>
<thead>
<tr>
<th></th>
<th>$D_1$</th>
<th>$D_2$</th>
<th>$D_3$</th>
<th>$D_4$</th>
<th>$D_5$</th>
<th>$D_6$</th>
<th>$S_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>7</td>
<td></td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
<td>13</td>
</tr>
<tr>
<td>$S_2$</td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>$S_3$</td>
<td>3</td>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>$S_4$</td>
<td>3</td>
<td>2</td>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>$D_1$</td>
<td>3</td>
<td>7</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>36</td>
</tr>
</tbody>
</table>

**TABLE 4.3 (a):** Shipment Matrix after one Iteration.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>$U_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>5</td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

**Table 4.3 (b):** Cost Matrix after one Iteration.
The values of the dual variables $U_i$ and $V_j$, for the new basis may be computed without reference to the cost data as follows:

Decrease the $U$ for the row of the added element by $\Delta C$. Now select the basic tree consisting of those elements of the basis which may be reached by basic paths from the added element which do not contain other elements in the column of the added element. Consider in turn the basic paths to the elements of the tree. If the last move along the path to an element is across a row, increase $V$ for its column by $\Delta C$; if the last move is up or down a column decrease the $U$ for the row of the element $\Delta C$.

In the example, the circled elements in Table 4.3(a) form the basic tree. The results are given in Table 4.3(b). New $U$'s and $V$'s again satisfy equations 4.1 for the elements of the new basis. The search for another element which, if added to the basis, would reduce the total cost may now be resummed to begin the next iteration.

When

$$\Delta C = U_i + V_j - C_{ij}$$  \hspace{1cm} (4.5)

is less than or equal to zero for all combinations of $i$ and $j$, the solution has been found.
CHAPTER 5

THE STEPPING STONE ALGORITHM

A set of X's which satisfies the restrictions of equations 2.3, 2.4, and 2.5 is called a feasible solution. If the set also minimizes C., it is an optimum feasible solution. One result of the theory of linear programming is that a feasible or optimum feasible solution to a transportation problem need contain no more than m + n - 1 non-zero X's\(^{16,22}\). The stepping stone method consists of:

1. Generating a feasible solution having no more than m + n - 1 non-zero X's. Such a solution is called a basic feasible solution.

2. Modifying the basic feasible solution to obtain a new basic feasible solution in a manner that will decrease the total cost.

3. Repeating step (2) until no further change will result in a decrease of cost.
The theory of the general simplex method proves that this process leads to a basic feasible solution which is an optimum feasible solution (12, 16).

Dantzig (5) shows that, for any basic feasible solution, numbers \(U_i\) and \(V_j\) can be found such that, for those \(X_{ij}\) in the basic solution, \(U_i + V_j = C_{ij}\). Dantzig also shows that, if \(U_i + V_j = \bar{C}_{ij}\) for those variables not in the basic solution and if all \(\bar{C}_{ij} - C_{ij} \leq 0\), then the basic feasible solution is also a minimum solution. If this condition of optimality is not satisfied, a new basic feasible solution can be readily obtained whose corresponding value of the objective function is less than (nondegeneracy assumed) the preceding value.

Dantzig's ingenious computational procedure enables one to obtain basic feasible solutions without setting up the usual simplex table and to test for optimality, i.e., to compute the \(Z_{ij} = \bar{C}_{ij}\), in terms of the basis vectors (12) without the explicit representation of the vectors not in the basis.

There are many methods of determining an initial basic feasible solution. It is worth-while to spend some time finding a good initial solution because it can considerably reduce the total number of iterations required to reach an optimal solution.

Most of the methods for determining an initial basic feasible solution assign a positive value to one variable and, at the same time, satisfy either a row or column constraint at each step.
NORTHWEST CORNER RULE

A particularly simple method of determining an initial basic feasible solution is the so-called Northwest Corner Rule, introduced by Charnes and Cooper (2).

Begin with cell (1,1). Set \( X_{11} = \min (S_1, D_1) \). At this first step, satisfy either an origin or a destination requirement.

If \( S_1 > D_1 \), move to cell (1,2) and set \( X_{12} = \min (S_1 - D_1, D_2) \). On the other hand, if \( D_1 > S_1 \), move to cell (2,1) and set \( X_{21} = \min (D_1 - S_1, S_2) \). When \( S_1 = D_1 \), degeneracy occurs, this will be treated later.

At the second step, satisfy either the second origin or the second destination requirement. Continue in this way, satisfying at the \( k^{th} \) step either an origin or a destination requirement. Ultimately, this results is a feasible solution.

This method cannot yield more than \( m + n - 1 \) positive \( X_{ij} \) because, at each step, an origin or a destination requirement is satisfied (16). After \( m + n - 1 \) steps, \( m + n - 1 \) of the constraints will be satisfied.

In the absence of degeneracy, there are not less than \( m + n - 1 \) positive \( X_{ij} \). In this case, it is clear that the resulting solution is a basic because the method of constructing the solution rules out any possibility of loops, and at each step, a row or a column constraint is satisfied. In constructing the solution, movement is always down and to the right, and hence a loop cannot be formed.
The basic feasible solution obtained by mean of the Northwest Corner rule may be far from optimal since the costs were completely ignored. Other methods of determining an initial basic feasible solution which do take account of the costs are considered in chapter 6. The additional effort spent in obtaining a good initial basic solution is worth while because it can considerably reduce the number of iterations which will be required to find an optimal solution.

The solutions obtained by the Northwest Corner rule (and similar schemes) are extreme-point solutions, and only such solutions need be considered as candidates for the minimum feasible solution.

Since virtually all applications of the transportation problem require the shipping of only whole units of the item being considered, it is useful to establish the following important property of the transportation problem(16).

Assuming the \( S_i \) and \( D_j \) are non-negative integers, then every basic feasible solution (i.e., extreme-point solution) has integral values. Any optimal basic solution to transportation problem will have the property that all positive \( X_{ij} \) will be integers.

This integrality property is peculiar to the transportation problem. In general, an optimal solution to an arbitrary linear programming problem may not have integral values for the variables. In fact, for the variables to be integers, a linear programming problem usually becomes a nonlinear programming problem(2,7). Intuitively, this integrality property is expected to follow the physical nature of the problem: if it
is profitable to ship a fraction of a unit to any destination, it is generally profitable to ship as large a quantity as possible. Since an integral number of units is required at each destination, an integral number of units will be shipped.

The reduced system of $m + n - 1$ equations in $mn$ variables can, of course, be solved by the general simplex procedure. However, for even small values of $m$ and $n$, the resulting system of equations becomes unwieldy for manual computation. This dilemma is resolved by the special adaptation of the simplex algorithm to the transportation problem.

EXAMPLE

Consider a problem involving 4 origins and 6 destinations. The origin availabilities, the destination requirements, and the costs are given in Table 5.1.

Note that $\sum s_i = \sum d_j = 181$

An initial basic feasible solution can be obtained by means of Northwest Corner Rule:

Set $x_{11} = \min (s_1, d_1) = \min (50, 30) = 30$
TABLE 5.1: Combined Allocation and cost Matrix

<table>
<thead>
<tr>
<th></th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>Dj</th>
<th>Vj</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>2</td>
<td>30</td>
<td>2</td>
<td>0</td>
<td>50</td>
<td>2</td>
</tr>
<tr>
<td>D2</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>D3</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>D4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>D5</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>D6</td>
<td>5</td>
<td>-1</td>
<td>-4</td>
<td>-2</td>
<td>2</td>
<td>-1</td>
</tr>
</tbody>
</table>

Cost Matrix:

<table>
<thead>
<tr>
<th></th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>Dj</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>2</td>
<td>50</td>
<td>40</td>
<td>60</td>
<td>31</td>
</tr>
<tr>
<td>Vj</td>
<td>3</td>
<td>181</td>
<td>11</td>
<td>30</td>
<td>1</td>
</tr>
</tbody>
</table>

Cost Values:

- S1: 2, 50, 40, 60, 31
- S2: 40, 50, 40, 60, 31
- S3: 50, 40, 50, 60, 31
- S4: 40, 60, 60, 60, 31
- Dj: 50, 50, 40, 30, 1

Notes:
- D1: 30, 20
- D2: 30, 30
- D3: 10, 10
- D4: 10, 10
- D5: 11, 11
- D6: 20, 20

27
Thus satisfying the requirements of destination 1. But there are still units available at origin, hence set:

\[ X_{12} = \min (S_1 - D_1, D_2) = \min (20, 50) = 20 \]

Now the first origin constraint is satisfied. Since an additional 30 units must be shipped to destination 2, set:

\[ X_{22} = \min (D_2 - 20, S_2) = \min (30, 40) = 30 \]

Thus the requirements of destination 2 are satisfied. 10 units are still available at origin 2. Thus:

\[ X_{23} = \min (10, D_3) = 10 \]

An additional 10 units must be shipped to destination 3, set: \( X_{23} = 10 \)

This leaves 50 units still to be shipped from origin 3. Hence set: \( X_{34} = 40 = D_4 \)

and satisfy the requirement of destination 4. Since 10 units remain to be shipped from origin 3, set: \( X_{35} = 10 \)

The requirement at destination 5 is 30, set: \( X_{45} = 20 \)

This leaves 11 units to be shipped from origin 5, which is precisely the number of units required at destination 6.
Hence: \[ x_{46} = 1. \]

Circle the \( x_{ij} \) values just obtained and note that they are 9 = \( m + n - 1 \) in number. Furthermore, these cells do not form a loop. This is a basic feasible solution. All other \( x_{ij} \) are zero, which are not filled in. The initial basic feasible solution is shown in Table 5.1.

Now determine, for those variables in the basic solution, \( m \) numbers of \( U_i \) and \( n \) numbers of \( V_j \) such that:

\[
\begin{align*}
U_1 + V_1 &= C_{11} = 2 \\
U_1 + V_2 &= C_{12} = 1 \\
U_2 + V_2 &= C_{22} = 2 \\
U_2 + V_3 &= C_{23} = 2 \\
U_3 + V_3 &= C_{33} = 4 \\
U_3 + V_4 &= C_{34} = 2 \\
U_3 + V_5 &= C_{35} = 4 \\
U_4 + V_5 &= C_{45} = 2 \\
U_4 + V_6 &= C_{46} = 2
\end{align*}
\]

(5.1)

Here there are 10 variables in 9 (i.e., \( m + n - 1 \)) equations. Since equation 5.1 is an underdetermined set of linear equations (i.e., the number of unknowns exceeds the number of equations), the system has an infinite number of solutions. A solution could be determined by letting any one of the variables equal its corresponding \( C_{ij} \). This reduces the number of unknowns by 1 and forces a unique solution of \( m + n - 1 \) equations in the remaining \( m + n - 1 \) variables.
Letting \( V_1 = 0 \), results in

\[
\begin{align*}
U_1 &= 2 & V_1 &= 0 \\
U_2 &= 3 & V_2 &= 1 \\
U_3 &= 5 & V_3 &= -1 \\
U_4 &= 3 & V_4 &= -3 \\
V_5 &= -1 \\
V_6 &= -1
\end{align*}
\]

(5.2)

The results are shown in Table 5.1, the corresponding cost coefficients are circled.

Since all the equations of 5.1 are satisfied, \( U_i + V_j = C_{ij} \) for those \( X_{ij} \) in the basic feasible solution. Now compute \( \bar{C}_{ij} = U_i + V_j \) for all combinations \( (i,j) \), \( \bar{C}_{ij} = C_{ij} \) for all \( X_{ij} \) in the solution).

The next task is to compute \( \bar{C}_{ij} - C_{ij} \). These are calculated and are shown in Table 5.1.

Not all \( \bar{C}_{ij} - C_{ij} \leq 0 \), so the initial basic feasible solution is not optimal. The largest \( \bar{C}_{ij} - C_{ij} \) is \( \bar{C}_{36} - C_{36} = 3 \). Hence select \( X_{36} \) to be introduced into the solution at an unknown non-negative level \( \Delta X \).

As the row and column sums of the variables must equal the corresponding values of \( S_i \) and \( D_j \), add or subtract \( \Delta X \) from some of the other \( X_{ij} \) in the first solution.
<table>
<thead>
<tr>
<th></th>
<th>D₁</th>
<th>D₂</th>
<th>D₃</th>
<th>D₄</th>
<th>D₅</th>
<th>D₆</th>
<th>S₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₁</td>
<td>30</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>S₂</td>
<td>30</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>S₃</td>
<td>10</td>
<td>40</td>
<td>10-ΔX</td>
<td>ΔX</td>
<td></td>
<td></td>
<td>60</td>
</tr>
<tr>
<td>S₄</td>
<td></td>
<td></td>
<td>20+ΔX</td>
<td>11-ΔX</td>
<td></td>
<td></td>
<td>31</td>
</tr>
<tr>
<td>D₆</td>
<td>30</td>
<td>50</td>
<td>20</td>
<td>40</td>
<td>30</td>
<td>11</td>
<td>181</td>
</tr>
</tbody>
</table>

**TABLE 5.2:** Introduction of ΔX

---

<table>
<thead>
<tr>
<th></th>
<th>D₁</th>
<th>D₂</th>
<th>D₃</th>
<th>D₄</th>
<th>D₅</th>
<th>D₆</th>
<th>S₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₁</td>
<td>30</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>S₂</td>
<td>30</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>S₃</td>
<td>10</td>
<td>40</td>
<td></td>
<td>10</td>
<td></td>
<td></td>
<td>60</td>
</tr>
<tr>
<td>S₄</td>
<td></td>
<td></td>
<td></td>
<td>30</td>
<td>1</td>
<td></td>
<td>31</td>
</tr>
<tr>
<td>D₆</td>
<td>30</td>
<td>50</td>
<td>20</td>
<td>40</td>
<td>30</td>
<td>11</td>
<td>181</td>
</tr>
</tbody>
</table>

**TABLE 5.3:** Shipment Matrix after one Iteration
Since \( X \geq 0 \) in cell (3,6) subtract \( \Delta X \) from \( X_{35}, X_{46} \) and add \( \Delta X \) to \( X_{45} \) in order to keep the row and column sums correct as shown in Table 5.2. The size of \( \Delta X \) is restricted by those \( X_{ij} \) from which it is subtracted. \( \Delta X \) cannot be larger than the smallest \( X_{ij} \) from which it is subtracted.

Here \( \Delta X \) must be greater than zero and less than or equal to 10 in order to preserve feasibility. Let \( \Delta X = 10 \). The new solution is shown in Table 5.3.

The objective function for this solution is equal to:

\[
382 - \max(\overline{C}_{ij} - C_{ij}) \geq 0 \cdot \Delta X = 382 - (3)(10) = 352
\]

The corresponding combined allocation and cost matrix table including new \((U_i, V_j)\) is given in Table 5.4.

Here \( \max(\overline{C}_{ij} - C_{ij}) \geq 0 \) = \( \overline{C}_{42} - C_{42} = \overline{C}_{43} - C_{43} = 3 \). Hence there is a tie. Cell (4,3) could be arbitrarily selected to enter the basis. Introducing \( \Delta X \geq 0 \) in cell (4,3) and adding and subtracting \( \Delta X \) from \( X_{ij} \) results in Tables 5.5 and 5.6.

The objective function becomes

\[
352 - \max(\overline{C}_{ij} - C_{ij}) \geq 0 \cdot \Delta X = 352 - (3)(1) = 349
\]

32
<table>
<thead>
<tr>
<th></th>
<th>$D_1$</th>
<th>$D_2$</th>
<th>$D_3$</th>
<th>$D_4$</th>
<th>$D_5$</th>
<th>$D_6$</th>
<th>$S_1$</th>
<th>$U_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>50</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>20</td>
<td>-2</td>
<td>-4</td>
<td>-4</td>
<td>-7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_2$</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>40</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>30</td>
<td>10</td>
<td>-4</td>
<td>-4</td>
<td>-5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_3$</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>60</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>-1</td>
<td>10</td>
<td>40</td>
<td>-3</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_4$</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>31</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>30</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_j$</td>
<td>30</td>
<td>50</td>
<td>20</td>
<td>40</td>
<td>30</td>
<td>11</td>
<td>181</td>
<td></td>
</tr>
<tr>
<td>$V_j$</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-3</td>
<td>-4</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE 5.4**: Combined Table after First Iteration
### TABLE 5.5: Introduction of $\Delta X$

<table>
<thead>
<tr>
<th></th>
<th>$D_1$</th>
<th>$D_2$</th>
<th>$D_3$</th>
<th>$D_4$</th>
<th>$D_5$</th>
<th>$D_6$</th>
<th>$S_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>30</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>$S_2$</td>
<td>30</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>$S_3$</td>
<td>10 $- \Delta X$</td>
<td>40</td>
<td></td>
<td>$10 + \Delta X$</td>
<td>60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_4$</td>
<td>$\Delta X$</td>
<td></td>
<td>30</td>
<td></td>
<td>1 $- \Delta X$</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>$D_j$</td>
<td>30</td>
<td>50</td>
<td>20</td>
<td>40</td>
<td>30</td>
<td>11</td>
<td>181</td>
</tr>
</tbody>
</table>

### TABLE 5.6: Shipment Matrix after Second Iteration

<table>
<thead>
<tr>
<th></th>
<th>$D_1$</th>
<th>$D_2$</th>
<th>$D_3$</th>
<th>$D_4$</th>
<th>$D_5$</th>
<th>$D_6$</th>
<th>$S_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>30</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>$S_2$</td>
<td>30</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>$S_3$</td>
<td>9</td>
<td>40</td>
<td></td>
<td>11</td>
<td>60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_4$</td>
<td>1</td>
<td></td>
<td>30</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_j$</td>
<td>30</td>
<td>50</td>
<td>20</td>
<td>40</td>
<td>30</td>
<td>11</td>
<td>181</td>
</tr>
</tbody>
</table>
The combined table with \((U_i, V_j)\) is shown in Table 5.7.

Here \([\max(\overline{C}_{ij} - C_{ij}) > 0] = \overline{C}_{31} - C_{31} = 2\)

Introduce \(\Delta X > 0\) in the cell \((3,1)\). Add and subtract \(\Delta X\) as shown in Table 5.8, selecting \(\Delta X = 9\) gives new solution as shown in Table 5.9.

The new objective function is:

\[
349 - [\max(\overline{C}_{ij} - C_{ij}) > 0] \cdot \Delta X = 349 - (2)(9)
\]
\[
= 331
\]

The new values of \((U_i, V_j)\) are shown in Table 5.10.

Here \([\max(\overline{C}_{ij} - C_{ij}) > 0] = \overline{C}_{44} - C_{44} = 1\).

Introduce \(\Delta X > 0\) in the cell \((4,4)\), add and subtract \(\Delta X\) as shown in Table 5.11. Selecting \(\Delta X = 1\) gives the new solution as shown in Table 5.12.

The new objective function is:

\[
331 - [\max(\overline{C}_{ij} - C_{ij}) > 0] \cdot \Delta X = 331 - (1)(1)
\]
\[
= 330
\]

The new values of \((U_i, V_j)\) are shown in Table 5.13.
Here all \(\overline{C}_{ij} - C_{ij} \leq 0\) and this last solution is a minimum feasible solution.
<table>
<thead>
<tr>
<th></th>
<th>$D_1$</th>
<th>$D_2$</th>
<th>$D_3$</th>
<th>$D_4$</th>
<th>$D_5$</th>
<th>$D_6$</th>
<th>$S_1$</th>
<th>$U_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>50</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>20</td>
<td>-2</td>
<td>-4</td>
<td>-1</td>
<td>-7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_2$</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>40</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>30</td>
<td>10</td>
<td>-4</td>
<td>-1</td>
<td>-5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_3$</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>60</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>-1</td>
<td>9</td>
<td>40</td>
<td>0</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_4$</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>31</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>30</td>
<td>-3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_J$</td>
<td>30</td>
<td>50</td>
<td>20</td>
<td>40</td>
<td>30</td>
<td>11</td>
<td>181</td>
<td></td>
</tr>
<tr>
<td>$V_J$</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-3</td>
<td>0</td>
<td>-4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE 5.7:** Combined Table after Second Iteration
### Table 5.8: Introduction of $\Delta X$

<table>
<thead>
<tr>
<th></th>
<th>$D_1$</th>
<th>$D_2$</th>
<th>$D_3$</th>
<th>$D_4$</th>
<th>$D_5$</th>
<th>$D_6$</th>
<th>$S_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>30 $-\Delta X$</td>
<td>20 $+\Delta X$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>$S_2$</td>
<td>30 $-\Delta X$</td>
<td>10 $+\Delta X$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>$S_3$</td>
<td></td>
<td>9 $-\Delta X$</td>
<td>40</td>
<td>11</td>
<td></td>
<td></td>
<td>60</td>
</tr>
<tr>
<td>$S_4$</td>
<td></td>
<td>1</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td>31</td>
</tr>
</tbody>
</table>

| $D_j$ | 30 | 50 | 20 | 40 | 30 | 11 | 181 |

### Table 5.9: Shipment Matrix after Third Iteration

<table>
<thead>
<tr>
<th></th>
<th>$D_1$</th>
<th>$D_2$</th>
<th>$D_3$</th>
<th>$D_4$</th>
<th>$D_5$</th>
<th>$D_6$</th>
<th>$S_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>21</td>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>$S_2$</td>
<td>21</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>$S_3$</td>
<td></td>
<td></td>
<td>40</td>
<td></td>
<td>11</td>
<td></td>
<td>60</td>
</tr>
<tr>
<td>$S_4$</td>
<td></td>
<td>1</td>
<td></td>
<td>30</td>
<td></td>
<td></td>
<td>31</td>
</tr>
</tbody>
</table>

| $D_j$ | 30 | 50 | 20 | 40 | 30 | 11 | 181 |
TABLE 5.10: Combined Table after Third Iteration

<table>
<thead>
<tr>
<th></th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>D5</th>
<th>D6</th>
<th>U1</th>
<th>V1</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>50</td>
<td>40</td>
<td>60</td>
<td>31</td>
<td>181</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>S2</td>
<td>-5</td>
<td>-1</td>
<td>-3</td>
<td>-2</td>
<td>-1</td>
<td>11</td>
<td>11</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>S3</td>
<td>-2</td>
<td>-1</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>10</td>
<td>40</td>
<td>1</td>
<td>40</td>
<td>1</td>
<td>40</td>
<td>1</td>
</tr>
<tr>
<td>S4</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>D1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>27</td>
<td>27</td>
<td>29</td>
<td>29</td>
<td>29</td>
<td>29</td>
<td>29</td>
</tr>
<tr>
<td>D2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>D3</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>D4</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>D5</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D6</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>U1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>V1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
### TABLE 5.11: Introduction of $\Delta X$

<table>
<thead>
<tr>
<th></th>
<th>$D_1$</th>
<th>$D_2$</th>
<th>$D_3$</th>
<th>$D_4$</th>
<th>$D_5$</th>
<th>$D_6$</th>
<th>$S_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>21-$\Delta X$</td>
<td>29+$\Delta X$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>$S_2$</td>
<td></td>
<td>21-$\Delta X$</td>
<td>19+$\Delta X$</td>
<td></td>
<td></td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>$S_3$</td>
<td>9 $X$</td>
<td></td>
<td></td>
<td>40-$\Delta X$</td>
<td></td>
<td>11</td>
<td>60</td>
</tr>
<tr>
<td>$S_4$</td>
<td></td>
<td></td>
<td>1-$\Delta X$</td>
<td>$\Delta X$</td>
<td>30</td>
<td></td>
<td>31</td>
</tr>
<tr>
<td>$D_J$</td>
<td>30</td>
<td>50</td>
<td>20</td>
<td>40</td>
<td>30</td>
<td>11</td>
<td>181</td>
</tr>
</tbody>
</table>

### TABLE 5.12: Shipment Matrix after Fourth Iteration

<table>
<thead>
<tr>
<th></th>
<th>$D_1$</th>
<th>$D_2$</th>
<th>$D_3$</th>
<th>$D_4$</th>
<th>$D_5$</th>
<th>$D_6$</th>
<th>$S_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>20</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>$S_2$</td>
<td></td>
<td>20</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>$S_3$</td>
<td>10</td>
<td></td>
<td>39</td>
<td></td>
<td>11</td>
<td></td>
<td>60</td>
</tr>
<tr>
<td>$S_4$</td>
<td></td>
<td>1</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td>31</td>
</tr>
<tr>
<td>$D_J$</td>
<td>30</td>
<td>50</td>
<td>20</td>
<td>40</td>
<td>30</td>
<td>11</td>
<td>181</td>
</tr>
<tr>
<td></td>
<td>$D_1$</td>
<td>$D_2$</td>
<td>$D_3$</td>
<td>$D_4$</td>
<td>$D_5$</td>
<td>$D_6$</td>
<td>$S_1$</td>
</tr>
<tr>
<td>---</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>$S_1$</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>(20)</td>
<td>(30)</td>
<td>-2</td>
<td>-2</td>
<td>0</td>
<td>-5</td>
<td></td>
</tr>
<tr>
<td>$S_2$</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>(20)</td>
<td>(20)</td>
<td>-2</td>
<td>0</td>
<td>-3</td>
<td></td>
</tr>
<tr>
<td>$S_3$</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>(10)</td>
<td>-3</td>
<td>-2</td>
<td>(39)</td>
<td>-1</td>
<td>(11)</td>
<td></td>
</tr>
<tr>
<td>$S_4$</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>-2</td>
<td>-1</td>
<td>-1</td>
<td>(1)</td>
<td>(30)</td>
<td>-2</td>
<td></td>
</tr>
<tr>
<td>$D_j$</td>
<td>30</td>
<td>50</td>
<td>20</td>
<td>40</td>
<td>30</td>
<td>11</td>
<td>181</td>
</tr>
<tr>
<td>$V_j$</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>-2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE 5.13:** Combined Table after Fourth Iteration
The stepping-stone algorithm permits the solution of transportation problems that are too large to lend themselves to direct application of the simplex method. For example, on a computer, it is not difficult to solve a transportation problem involving 1,000 destinations and 50 origins by means of the stepping-stone algorithm or some of its variants.

The stepping-stone algorithm has another property which is very important when a digital computer is to be used: it requires only the arithmetic operations of addition and subtraction. A digital computer can be made to operate in fixed-point arithmetic so that no rounding off errors occur in addition or subtraction. Thus, the stepping-stone algorithm makes it possible to avoid the problem of rounding-off errors which limit the size of the problems that can be solved by the simplex method. A transportation problem may require an arbitrarily large number of iterations, and no loss of accuracy due to rounding-off will occur if fixed-point arithmetic is used.
CHAPTER 6

DETERMINATION OF AN INITIAL BASIC FEASIBLE SOLUTION

The Northwest-Corner rule for determining an initial basic feasible solution to a transportation problem has already been discussed. Now some other methods which often yield a result much closer to an optimal solution than that obtained by the Northwest-Corner rule will be presented. As suggested before, it is worth while to spend some time finding a good initial solution because it can considerably reduce the total number of iterations required to reach an optimal solution.

Most of the methods for determining an initial basic feasible solution assign a positive value to one variable and, at the same time, satisfy either a row or column constraint at each step. Any procedure for determining a feasible solution which assigns a positive value to one variable and satisfies either a row or column constraint at each step will automatically yield a basic feasible solution, and in the absence of degeneracy, the resulting cells will form a basic tree. Such a technique cannot give more than \( m + n - 1 \) positive variables since, after \( m + n - 1 \) steps, \( m + n - 1 \) of the constraints will be satisfied, and the remaining constraint is automatically satisfied\(^{16}\). All methods described below make some use of the costs.
(1) COLUMN MINIMA.

Beginning with column 1 of the Table 2.5, choose the minimum cost in this column. Suppose that it occurs in row \( r \). Then set \( X_{r1} = \min(S_r, D_1) \). If \( X_{r1} = D_1 \), cross off column 1 and move to column 2. If \( X_{r1} = S_r \), cross off row \( r \) from the table, and choose the next lowest cost in column 1. Assume that it occurs in row \( s \). Set \( X_{s1} = \min(S_s, D_1 - S_r) \). Continue in this way until the requirement at the first destination is satisfied. If the minimum cost is not unique, select any one of the minima. When the requirement of column 1 is satisfied, cross off column 1 and repeat the above procedure for column 2. Continue until the requirement of column \( n \) is satisfied.

In the event that a row constraint and a column constraint, say column \( k \), are satisfied simultaneously, cross off only the row. Then move to the cell in column \( k \) having the next lowest cost. Assign a value of zero to this cell and assume it to be in the basic solution. Now cross off column \( k \) and move to column \( k + 1 \). This will yield a degenerate basic feasible solution.

If this procedure is used to obtain an initial basic solution for the example solved in Table 5.1, Table 6.1 is obtained, provided that in columns 2 and 3 the row with the lowest index is chosen when the minimum cost is not unique.
<table>
<thead>
<tr>
<th></th>
<th>$D_1$</th>
<th>$D_2$</th>
<th>$D_3$</th>
<th>$D_4$</th>
<th>$D_5$</th>
<th>$D_6$</th>
<th>$S_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>20</td>
<td>30</td>
<td>10</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>$S_2$</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_3$</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td></td>
<td>19</td>
<td>30</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_4$</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>30</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_j$</td>
<td>30</td>
<td>50</td>
<td>20</td>
<td>40</td>
<td>30</td>
<td>11</td>
<td>181</td>
</tr>
</tbody>
</table>

**TABLE 6.1:** Example of Column Minima.
(2) **ROW MINIMA:**

Beginning with row 1, choose the minimum cost in this row. Suppose that it occurs in column r. Set \( X_{1r} = \min(S_1, D_r) \). If \( X_{1r} = S_1 \), cross off row 1 and move to row 2. If \( X_{1r} = D_r \), cross off column r and determine the next lowest cost in row 1. Assume it occurs in column s. Set \( X_{1s} = \min(S_1 - D_r, D_s) \). Continue in this way until the first row constraint is satisfied. When the requirement of the first row is satisfied, cross off row 1 and repeat the above procedure for row 2. Continue until the row constraint \( m \) is satisfied. Whenever a minimum cost is not unique, make an arbitrary choice among the minima.

In the event that a row constraint, say row k, and a column constraint are satisfied simultaneously, cross off only the column. Then find the next lowest cost in row k, and insert this cell into the solution at a zero level. Then cross off row k, and move on to row \( k + 1 \).

If this technique is used to obtain a first solution for the problem of Table 6.1, Table 6.2 results. Note that degeneracy appears. According to the rule, either cell \((1,1)\) or \((1,5)\) could have been added at a zero level. Cell \((1,1)\) is chosen.

(3) **MATRIX MINIMA.**

Determine the smallest cost in the entire table. Suppose this occurs for cell \((i,j)\). Set \( X_{ij} = \min(S_i, D_j) \). Then cross off either
<table>
<thead>
<tr>
<th></th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$D_f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_1$</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>30</td>
</tr>
<tr>
<td>$D_2$</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>$D_3$</td>
<td></td>
<td></td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>$D_4$</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>$D_5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>$D_6$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Example of Row Minima.**

TABLE 6.2:
<table>
<thead>
<tr>
<th></th>
<th>D₁</th>
<th>D₂</th>
<th>D₃</th>
<th>D₄</th>
<th>D₅</th>
<th>D₆</th>
<th>S₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₁</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>.3</td>
<td>2</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>S₂</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>40</td>
</tr>
<tr>
<td>S₃</td>
<td>10</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>60</td>
</tr>
<tr>
<td>S₄</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>31</td>
</tr>
<tr>
<td>D₄</td>
<td>30</td>
<td>50</td>
<td>20</td>
<td>40</td>
<td>30</td>
<td>11</td>
<td>181</td>
</tr>
</tbody>
</table>

**TABLE 6.3:** Example of Matrix Minima.
row i or column j, depending on which requirement is satisfied. If $X_{ij} = S_j$, decrease $D_j$ by $S_i$, and if $X_{ij} = D_j$, decrease $S_i$ by $D_j$. Repeat the process for the resulting table. Whenever the minimum cost is not unique, make an arbitrary choice among the minima. If a row and a column constraint are satisfied simultaneously, cross off only the row or the column, not both.

This method yields the initial solution (shown in Table 6.3) for the example under consideration if, in the absence of a unique minimum, the cell for which $i + j$ is smallest is chosen. Here again degeneracy appears.

(4) Vogel's Method.

This technique has been suggested by Vogel\(^{23}\). For each row, find the lowest cost $C_{ij}$ and the next lowest cost $C_{it}$ in that row. Compute $C_{it} - C_{ij}$. In this way, m numbers are obtained. Proceed in the same way for each of the columns and obtain n more numbers. Choose the largest of these $m + n$ differences. Suppose that the largest of these numbers was associated with the difference in column j. Let cell $(i,j)$ contain the lowest cost in column j. Then set $X_{ij} = \min(S_i,D_j)$. Cross off either row i or column j, depending on which requirement is satisfied, and repeat the whole process for the resulting table. When the maximum difference is not unique, an arbitrary choice can be made, and if a row and a column constraint are satisfied simultaneously, cross off only the row or the column, not both.
TABLE 6.4: Example of Vogel's Method.

<table>
<thead>
<tr>
<th></th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>D5</th>
<th>D6</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>50</td>
<td>40</td>
<td>60</td>
<td>31</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>2</td>
<td>30</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>30</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D2</td>
<td>1</td>
<td>2</td>
<td>10</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3</td>
<td></td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D4</td>
<td></td>
<td></td>
<td>10</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D5</td>
<td></td>
<td></td>
<td></td>
<td>50</td>
<td>40</td>
<td>30</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

49
For the above example, this method yields the solution shown in Table 6.4. It is convenient to list the row differences in a column to the right of the table and the column differences in a row at the bottom of the table. The differences shown in the difference row and column are those for the first step, i.e., those which are to be used in selecting the first basis cell. This is the worst possible case; every difference has the same value. The tie is resolved by choosing the cell with the smallest value of \( i + j \). At each step, a new set of differences must be computed.

**CHOICE OF METHOD.**

Unfortunately, there is no easy means at present for determining an initial basic feasible solution which would lead to the smallest number of iterations\(^{(16)}\). It would be necessary to solve the problem in each case.
CHAPTER 7

DEGENERACY AND THE TRANSPORTATION PROBLEM

A feasible solution to a transportation problem is degenerate if less than \( m + n - 1 \) of the \( x_{ij} \) are positive. Degeneracy may be encountered in the process of determining the initial basic feasible solution or at some subsequent iteration. From the practical point of view, degeneracy does not cause any difficulties. No transportation problem has ever been known to cycle. The degeneracy problem can be eliminated by the use of a perturbation method as in the case of simplex method. However, a much simpler perturbation method can be used.

DEGENERACY DUE TO INSUFFICIENT POSITIVE \( x_{ij} \)'s:

If \( h \) is the number of \( x_{ij} > 0 \) and if the method used to provide an initial solution yields a feasible solution with \( h < m + n - 1 \) degeneracy occurs. The cells associated with the positive \( x_{ij} \) do not form a basic tree. To obtain an initial basic solution and a basic tree in the table, \( m + n - 1 - h \) additional cells at a zero level must be added. Choose the cells to be added such that the resulting \( m + n - 1 \) cells form a basic tree. Enter a zero into the cells added and circle these zeros to indicate that they are part of the initial basic solution. In manual computations,
<table>
<thead>
<tr>
<th></th>
<th>$D_1$</th>
<th>$D_2$</th>
<th>$D_3$</th>
<th>$D_4$</th>
<th>$D_5$</th>
<th>$D_6$</th>
<th>$S_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>25</td>
</tr>
<tr>
<td>$S_2$</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>25</td>
</tr>
<tr>
<td>$S_3$</td>
<td></td>
<td></td>
<td>40</td>
<td>10</td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>$S_4$</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>$S_5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>$D_j$</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td>10</td>
<td>20</td>
<td>170</td>
</tr>
</tbody>
</table>

**TABLE 7.1:** Degeneracy where $h < m + n - 1$ of Positive $X_{ij}$
it is very easy to choose cells which will yield a basic tree. Having determined a basic solution, proceed in the usual way. In this case a variable can enter and leave the solution at a zero level.

Consider the Table 7.1. Costs are omitted since they are not relevant to the discussion. The northwest-corner rule to find an initial feasible solution results in Table 7.1. Here there are only 8 positive \( X_{ij} \), although \( m + n - 1 = 10 \). It will be noted that the set of cells corresponding to the positive \( X_{ij} \), is not connected.

Two more cells are needed to obtain a basic tree. Adding cells (2,3) and (4,5) (dashed circles), a basic tree results. The value zero is entered into these cells, giving a basic (degenerate) feasible solution.

If the \( \bar{c}_{ij} - c_{ij} \) are such that cell (2,4) should appear in the basic tree at the next iteration, \( X_{24} \) will enter the basic solution at a zero level. The values of the variables in the basic solution remain unchanged. Only cell (2,3) is replaced by cell (2,4) to obtain a new basic tree.

DEGENERACY DUE TO TIE OF VARIABLES:

Degeneracy can also appear at some later iteration if there is a tie for the variable to leave the basic solution. Choose arbitrarily any one of the tied variables as the variable to leave the basis. At the next stage, the variables that were tied with the removed variable will be at a zero level. However, keep these variables in the basic solution (i.e.,
<table>
<thead>
<tr>
<th></th>
<th>$D_1$</th>
<th>$D_2$</th>
<th>$D_3$</th>
<th>$D_4$</th>
<th>$D_5$</th>
<th>$D_6$</th>
<th>$S_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>30</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>$S_2$</td>
<td>30</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>$S_3$</td>
<td>10</td>
<td>40</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td>60</td>
</tr>
<tr>
<td>$S_4$</td>
<td></td>
<td></td>
<td></td>
<td>20</td>
<td>10</td>
<td></td>
<td>30</td>
</tr>
<tr>
<td>$D_J$</td>
<td>30</td>
<td>50</td>
<td>20</td>
<td>40</td>
<td>30</td>
<td>10</td>
<td>180</td>
</tr>
</tbody>
</table>

**TABLE 7.2:** Initial Basic Feasible Solution  
(with tie of Variables)
<table>
<thead>
<tr>
<th></th>
<th>$D_1$</th>
<th>$D_2$</th>
<th>$D_3$</th>
<th>$D_4$</th>
<th>$D_5$</th>
<th>$D_6$</th>
<th>$S_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>30</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>$S_2$</td>
<td>30</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>$S_3$</td>
<td>10</td>
<td>40</td>
<td></td>
<td></td>
<td>10</td>
<td></td>
<td>60</td>
</tr>
<tr>
<td>$S_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>30</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>$D_J$</td>
<td>30</td>
<td>50</td>
<td>20</td>
<td>40</td>
<td>30</td>
<td>10</td>
<td>180</td>
</tr>
</tbody>
</table>

**TABLE 7.3**: Solution of Table 7.2, Degeneracy occurred
they remain circled) and proceed as usual.

Suppose that in the problem in Table 5.1 the availability at origin is changed from 4 to 30 and the requirement at destination 6 to 10. For this new problem, the northwest-corner rule yields the initial basic feasible solution shown in Table 7.2.

The $\bar{c}_{ij} - c_{ij}$ are the same as in Table 5.1, so $x_{36}$ enters the basic solution at the next stage. Now, there is a tie for the variable to be removed. Either $x_{35}$ or $x_{46}$ can be replaced by $x_{36}$. If $x_{35}$ is arbitrarily replaced, the new basic solution is that shown in Table 7.3. One of the basic variables is now zero, and degeneracy has appeared. Again, the $\bar{c}_{ij} - c_{ij}$ are the same as in Table 5.4. If $x_{43}$ is chosen to enter the basis, it enters at a zero level.

Suppose that at step $k$, a row and a column constraint are satisfied simultaneously, this means that degeneracy has appeared. Now, imagine that either the row or column constraint satisfied at step $k$ is perturbed by increasing its requirement by $\Delta$. Then continue the process of finding the initial solution. Hadley (16) has shown that the resulting solution will be basic, and when $\Delta$'s are set to zero, a degenerate basic solution is found. It is really unnecessary to introduce $\Delta$'s explicitly. It eliminates the necessity of introducing numerical values of $\Delta$'s, when problem is solved with digital computers. Thus cells at zero level can be automatically added.
CHAPTER 8

MODIFIED STEPPING STONE ALGORITHM

FOR DIGITAL COMPUTERS

Associated with each basic feasible solution is a basis whose elements include all positions in the shipment matrix where \( X_{ij} \) does not equal zero. The following procedure may be used to form an initial basic feasible solution and the corresponding basis with which to start the stepping stone method:

Examine the entries in the first row of the cost matrix and select the entry having the smallest cost. Include this position in the shipment matrix as an element of the basis, with its value equal to the smaller of the supply for its row and the demand for its column. Decrease the supply and demand by this amount. If a positive supply is left, drop the column whose demand was just satisfied from further consideration and again look for the lowest cost in this row. If a positive demand is left, insert an element in the same column but the next row, and again check whether there is supply or demand left over, and proceed as above. This is the row minima method described in Chapter 6.

This process yields in general a basis containing exactly \( m + n - 1 \).
elements with no loops. An example of a shipment matrix and a basic feasible solution generated in this manner from given supply and demand data is shown in Table 8.2 and 8.3.

**DATA STORAGE IN THE COMPUTER:**

The input data required to specify a transportation problem are:

1. cost matrix $C_{ij}$: $(mn)$ values,
2. supplies and demands $S_i$ and $D_j$: $(m + n)$ values.

After the formation of the initial basic solution, the data required in the performance of the iterations are:

1. shipment matrix $X_{ij}$: $(m + n - 1)$ values,
2. dual variables $U_i$ and $V_j$: $(m + n)$ values,
3. cost matrix $C_{ij}$: $(mn)$ values.

The largest volume of data that must be handled is the cost data. However, it is only necessary to have access to this data sequentially. Therefore, it may be recorded on some slow access, high capacity storage medium like magnetic tape or magnetic drum.

**MAJOR COMPUTER ROUTINES:**

The block diagram of the major computer routines required for the transportation problem is shown in figure 8.1.

**Part 1 and 8:**

The input and output blocks are not of interest because they involve no unusual ideas.
Fig. 8.1: MAJOR ROUTINES

1. The flow charts are given in Appendix A except for Part 1 and 8 which are user dependent.

2. The program listing is given in Appendix B.
Table 8.2: Shipment Matrix

<table>
<thead>
<tr>
<th></th>
<th>D₁</th>
<th>D₂</th>
<th>D₃</th>
<th>D₄</th>
<th>D₅</th>
<th>D₆</th>
<th>S₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₁</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>S₂</td>
<td></td>
<td></td>
<td>4</td>
<td>5</td>
<td>6</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>S₃</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>S₄</td>
<td>1</td>
<td>8</td>
<td>3</td>
<td>9</td>
<td></td>
<td></td>
<td>21</td>
</tr>
<tr>
<td>S₅</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>D₆</td>
<td>1</td>
<td>10</td>
<td>3</td>
<td>11</td>
<td>14</td>
<td>16</td>
<td>55</td>
</tr>
</tbody>
</table>

Table 8.3: Basis Table

<table>
<thead>
<tr>
<th>t</th>
<th>Iₜ</th>
<th>Kₜ</th>
<th>Jₜ</th>
<th>Lₜ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>8</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>7</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>9</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>10</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>
Part 2:

The "Form initial basic feasible solution" block sets up a feasible solution in the manner described above, computes K and L values for the basis table, and calculates the initial values of the dual variables \( U_i \) and \( V_j \).

Part 3:

This block searches through the cost data in sequence and selects a position in the shipment matrix for which \( U_i + V_j - C_{ij} \) is positive as the next element to be added to the basis. If no such position is found, the present basis is an optimum solution and the results are printed. The amount of cost data which is examined for each iteration by this block has a very important effect on the overall time requirement for solving a problem.

Part 4 and 5:

The next two blocks perform the tasks of finding the basic loop including the new element, finding the element that limits the size of the new element and hence drops from the basis, and changing the values of the elements in the loop.

Part 6:

The "Modify basis table" block makes the necessary changes in the K and L values of the basis table to take care of the new element. It also shifts an entry in the table, if required, to maintain the ordering of the first \( n \) entries.
Part 7:

Block seven computes the U's and V's for the new basis. The details of each routine (except Part 1 and 8) are given in Appendix A and a program listing in Appendix B. Details for Part 1 and 8 (input/output blocks) are not given as they are user dependent.

DETAILS OF THE PROCEDURE:

From the discussion of the stepping stone method it is seen that the shipment matrix must be referred essentially at random. Therefore, in order to facilitate high speed operation, the shipment matrix must be stored in the high speed memory of the computer. Since, at most \( m + n - 1 \) of the positions of the shipment matrix can contain non-zero \( X_{ij} \)'s, a basic feasible solution may be stored as a table of \( m + n - 1 \) entries giving the row, column and value of each element of the basis.

In carrying out the processes of the stepping stone method, basic paths between elements must be traced. This requires much referring on the part of the computer between basic elements either in the same column or the same row of the shipment matrix. In order to allow the computer to proceed as rapidly as possible in tracing the paths, the following information may be stored for each entry in the basis table.

1. Data informing the computer where to find entries for the other basic elements in the same row of the shipment matrix.
(2) Data informing the computer where to find entries for the other basic elements in the same column of the shipment matrix.

One convenient way of representing this information is shown in Table 8.3. For entry number $t$ of the table, the value of $K_t$ gives the entry number of another basic element in the same row; the value of $L_t$ gives the entry number of a basic element in the same column of the shipment matrix. The table is arranged so that by jumping from entry to entry as directed by $K$ (or $L$) values, all basic elements in a particular row (column) of the shipment matrix will be encountered. Also, for ease in finding the entry for a basic element in a particular column, the first $n$ entries of the basis table are for elements in columns of the shipment matrix corresponding to their entry number. These features of the basis table give the program to be described its high speed. However, the arrangement has the slight disadvantage that as the basis changes with each iteration, the new basis table must be calculated.

The numbers in the shipment matrix are the entry numbers of the elements in the basis table and not the values of the elements. Each point on the graph is identified by the entry number of the element it represents in the basis table. The graph starts with the element in the basis table whose entry number is the column number of the element being introduced. This element will always be in the same column of the shipment matrix as the new element, i.e., column 1 in the example. A move from left to right in the graph corresponds to moving from one element in a given row of the shipment matrix to the element in the same row designated by the $K$ value of the first in the basis table. Similarly, a
move down in the graph corresponds to moving to the element in the same
column of the shipment matrix designated by the L value in the basis
table.

In searching for the loop involving the new element, start from an
element in the same column as the new element. Hence a basic path from
the starting element to an element in the same row as the new element
must be found. This is done by searching the branches of the graph
successively until an element in the correct row is found. Sufficient
information must be remembered during the search so that the path may be
identified once it is found. The search is carried out as follows:
First, a path is traced through the tree without turning at any of the
branch points as shown by the arrow labelled 1 in figure 8.4. While
tracing this path the branch points encountered are as in Table 8.5 (b)
to provide starting points for searching the remaining branches of the
table. Entry b in the branch point table gives the basis table entry
number of the branch point $G_b$ and an indicator $B_b$ telling whether the
branch point was encountered while moving across a row ($B_b = 0$) or down
a column ($B_b = 1$). A separate tabulation, Table 8.5 (a) is kept of
those elements (branch points and corner points) which could belong to
the desired loop. Entry a in the search table gives the basis table
entry number $F_a$ of the element, and tells whether it is a corner point
($A_a = 0$) or branch point ($A_a = 1$) of the graph.

Since an entry in row 3 was not found while searching this path,
the last entry in the branch point table is examined and the search is
continued by tracing the path indicated by the arrow labelled 2 in figure
8.4. The number of this branch point is recorded in the search table and
Fig. 8.4: Sequence of Searching the Branches of a Tree

<table>
<thead>
<tr>
<th>a</th>
<th>F_a</th>
<th>A_a</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>1</td>
</tr>
</tbody>
</table>

(a) Search Table

<table>
<thead>
<tr>
<th>b</th>
<th>G_b</th>
<th>B_b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>0</td>
</tr>
</tbody>
</table>

(b) Branch Point Table

Table 8.5: First Stage of the Search for a Loop
identified by $A_a = 2$ to denote that a new path is being traced as shown in Table 8.6. The search is continued as indicated in figure 8.4 until element 7 is reached which is in row 3. Since element 7 is an element of the desired loop, it is entered in the search table. The state of the tables is then as in Table 8.7, and the search is terminated.

The decisions as to whether a particular element $t$ is a branch point or a corner point, and what basic element is to be examined next, may be easily made by examining $K$ and $L$ values in the basis table. For instance, suppose we have just moved to element $t$ from another element in the same row. Let $h$ be the entry number of the first element that was examined in this row. Then,

(a) if $K_t = h$ and $L_t = t$, the end of this path has been reached.

(b) if $k_t = h$ and $L_t \neq t$, element is a corner point; examine element number $L_t$ next.

(c) if $K_t \neq h$ and $L_t = t$, element $t$ is neither a branch or corner point; examine element number $K_t$ next.

(d) if $K_t \neq h$ and $L_t \neq t$, element is a branch point; examine element number $K_t$ next.

Completely analogous rules apply for moving an element in the same column of the shipment matrix.

The determination of the elements of the basic loop from the entries in the search table may be accomplished by considering the entries
### Table 8.6: Second Stage of Search

<table>
<thead>
<tr>
<th>a</th>
<th>$F_a$</th>
<th>$A_a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>0</td>
</tr>
</tbody>
</table>

### Table 8.7: Completion of Search

<table>
<thead>
<tr>
<th>a</th>
<th>$F_a$</th>
<th>$A_a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>0</td>
</tr>
</tbody>
</table>

### Table (b) Branch Point Table

<table>
<thead>
<tr>
<th>b</th>
<th>$G_b$</th>
<th>$B_b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>b</th>
<th>$G_b$</th>
<th>$B_b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
<td>0</td>
</tr>
</tbody>
</table>

of the search table in sequence starting with the last and applying the following rules:

1. The last element in the search table is the first element of the loop.

2. If entry \(a\) of the search table has \(A_a = 0\), element \(F_a\) is in the loop.

3. If entry \(a\) of the search table has \(A_a = 1\), it is ignored except as noted below.

4. If entry \(a\) of the search table has \(A_a = 2\), all entries are ignored until an entry \(b\) is found with \(A_b = 1\), \(F_b = F_a\). Element \(F_a\) is a member of the loop.

Essentially the identical search technique is used in calculating the new values of the dual variables. The difference is that the object of the search is to examine all of the elements in a particular sub-tree of the basis rather than to find an element in a particular row.

When the logical search procedure is used on a high speed computer, the time required to carry out the operations of one iteration becomes short compared to the time required to search through the entire cost data in finding the new element to be put in the basis. Thus, the total time for solution might be cut down if the time spent searching the cost data per iteration were shortened at the expense of a larger number of
iterations.

The average time required to perform an iteration using the method described here depends linearly on the size of the problem, $m + n$. This is a consequence of the fact that in searching for a loop, each entry in the basis table is considered no more than once. Since experience has shown that the number of iterations required to solve a problem is roughly proportional to $m + n$, the overall time requirements should increase as the square of the size of the problem. All this indicates the feasibility of solving extremely large transportation type problems in a economical amount of time.

The example of Chapter 5 was solved by this technique. The results are shown below.
**NUMERO OF ROWS** \( M = 4 \)  
**NUMBER OF COLS** \( N = 6 \)  

\[
\begin{array}{cccccc}
30 & 50 & 20 & 40 & 30 & 11 \\
50 & 40 & 60 & 31 \\
\end{array}
\]

**ITERATION 1**  
**NUMBER OF ROWS** \( M = 4 \)  
**NUMBER OF COLS** \( N = 6 \)

**BASIS TABLE**

<table>
<thead>
<tr>
<th>I(KT)</th>
<th>J(KT)</th>
<th>L(KT)</th>
<th>S(KT)</th>
<th>X(KT)</th>
<th>F(KT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**SHIPMENT MATRIX**

| 50 | 0 | 50 | 0 | 0 | 0 |
| 40 | 0 | 10 | 0 | 30 | 0 |
| 60 | 0 | 30 | 0 | 11 | 0 |
| 31 | 0 | 50 | 0 | 11 | 0 |

| 181 | 30 | 50 | 20 | 40 | 30 | 11 |

**COST MATRIX**

| 0 | 0 | 1 | 0 | 0 | 2 | 0 |
| 1 | 0 | 0 | 2 | 0 | 3 | 0 |
| 3 | 0 | 4 | 2 | 0 | 1 | 0 |
| 2 | 4 | 0 | 1 | 0 | 0 | 0 |

| 410 | 2 | 1 | 1 | -1 | 2 | -2 |

**TOTAL COST** 410

**ELEMENT MATRIX**

| 1 | 0 | 2 | 0 | 0 | 5 | 0 |
| 2 | 0 | 3 | 0 | 7 | 0 |
| 3 | 0 | 8 | 4 | 0 | 6 |
| 4 | 1 | 0 | 9 | 0 | 0 |

70
**ITERATION 2**

**NUMBER OF ROWS** \( M = 4 \)

**NUMBER OF COLS** \( N = 6 \)

**BASIS TABLE**

<table>
<thead>
<tr>
<th>1</th>
<th>3</th>
<th>6</th>
<th>1</th>
<th>0</th>
<th>30</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>50</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>7</td>
<td>3</td>
<td>2</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>8</td>
<td>6</td>
<td>0</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>9</td>
<td>4</td>
<td>0</td>
<td>31</td>
<td>0</td>
</tr>
</tbody>
</table>

**ELEMENTS ADDED** 12 = 3

**J2 = 1**

**ELEMENTS DROPPED** 11 = 4

**J1 = 1**

**IH1 = 1**

**SHIPMENT MATRIX**

<table>
<thead>
<tr>
<th>* 50&lt;sup&gt;+&lt;/sup&gt;</th>
<th>0&lt;sup&gt;+&lt;/sup&gt;</th>
<th>50&lt;sup&gt;+&lt;/sup&gt;</th>
<th>0&lt;sup&gt;+&lt;/sup&gt;</th>
<th>0&lt;sup&gt;+&lt;/sup&gt;</th>
<th>0&lt;sup&gt;+&lt;/sup&gt;</th>
<th>0&lt;sup&gt;+&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>* 40&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>10&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>30&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
</tr>
<tr>
<td>* 60&lt;sup&gt;+&lt;/sup&gt;</td>
<td>30&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>10&lt;sup&gt;+&lt;/sup&gt;</td>
<td>9&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>11&lt;sup&gt;+&lt;/sup&gt;</td>
</tr>
<tr>
<td>* 31&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>31&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
</tr>
<tr>
<td>* 181&lt;sup&gt;+&lt;/sup&gt;</td>
<td>30&lt;sup&gt;+&lt;/sup&gt;</td>
<td>50&lt;sup&gt;+&lt;/sup&gt;</td>
<td>20&lt;sup&gt;+&lt;/sup&gt;</td>
<td>40&lt;sup&gt;+&lt;/sup&gt;</td>
<td>30&lt;sup&gt;+&lt;/sup&gt;</td>
<td>11&lt;sup&gt;+&lt;/sup&gt;</td>
</tr>
</tbody>
</table>

**COST MATRIX**

<table>
<thead>
<tr>
<th>* -2&lt;sup&gt;+&lt;/sup&gt;</th>
<th>0&lt;sup&gt;+&lt;/sup&gt;</th>
<th>1&lt;sup&gt;+&lt;/sup&gt;</th>
<th>0&lt;sup&gt;+&lt;/sup&gt;</th>
<th>0&lt;sup&gt;+&lt;/sup&gt;</th>
<th>2&lt;sup&gt;+&lt;/sup&gt;</th>
<th>0&lt;sup&gt;+&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>* -1&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>2&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>3&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
</tr>
<tr>
<td>* 1&lt;sup&gt;+&lt;/sup&gt;</td>
<td>3&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>4&lt;sup&gt;+&lt;/sup&gt;</td>
<td>2&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>1&lt;sup&gt;+&lt;/sup&gt;</td>
</tr>
<tr>
<td>* 0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>1&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
</tr>
<tr>
<td>* 350&lt;sup&gt;++&lt;/sup&gt;</td>
<td>2&lt;sup&gt;+&lt;/sup&gt;</td>
<td>3&lt;sup&gt;+&lt;/sup&gt;</td>
<td>3&lt;sup&gt;+&lt;/sup&gt;</td>
<td>1&lt;sup&gt;+&lt;/sup&gt;</td>
<td>4&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
</tr>
</tbody>
</table>

**TOTAL COST** 350

**ELEMENT MATRIX**

<table>
<thead>
<tr>
<th>* 1&lt;sup&gt;+&lt;/sup&gt;</th>
<th>0&lt;sup&gt;+&lt;/sup&gt;</th>
<th>2&lt;sup&gt;+&lt;/sup&gt;</th>
<th>0&lt;sup&gt;+&lt;/sup&gt;</th>
<th>0&lt;sup&gt;+&lt;/sup&gt;</th>
<th>5&lt;sup&gt;+&lt;/sup&gt;</th>
<th>0&lt;sup&gt;+&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>* 2&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>3&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>7&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
</tr>
<tr>
<td>* 3&lt;sup&gt;+&lt;/sup&gt;</td>
<td>1&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>8&lt;sup&gt;+&lt;/sup&gt;</td>
<td>4&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>6&lt;sup&gt;+&lt;/sup&gt;</td>
</tr>
<tr>
<td>* 4&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>9&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
<td>0&lt;sup&gt;+&lt;/sup&gt;</td>
</tr>
</tbody>
</table>
ITERATION 3  *** NUMBER OF ROWS  -M =  4  *** NUMBER OF COLS  -N =  6

**** BASIS TABLE ****

<table>
<thead>
<tr>
<th>K</th>
<th>I(K)</th>
<th>K(K)</th>
<th>J(K)</th>
<th>L(K)</th>
<th>V(K)</th>
<th>F(K)</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>30</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>50</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>7</td>
<td>3</td>
<td>0</td>
<td>20</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>19</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>0</td>
<td>11</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>20</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>9</td>
<td>5</td>
<td>1</td>
<td>10</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>0</td>
<td>21</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

ELEMENTS ADDED  I2 =  4  J2 =  5
ELEMENTS DROPPED  I1 =  3  J1 =  3  IH1 =  A

**** SHIPMENT MATRIX ****

* 50**  0+  50+  0+  0+  0+  0+
* 40**  0+  0+  20+  0+  20+  0+
* 60**  30+  0+  0+  19+  0+  11+
* 31**  0+  0+  0+  21+  10+  0+
* 181**  30+  50+  20+  40+  30+  11+

**** COST MATRIX ****

* -28*  0+  1+  0+  0+  2+  0+
* -18*  0+  0+  2+  0+  3+  0+
* -18*  3+  0+  0+  2+  0+  1+
* -28*  0+  0+  0+  1+  2+  0+
* 330**  4+  3+  3+  3+  4+  2+

TOTAL COST  330

**** ELEMENT MATRIX ****

* 18*  0+  2+  0+  0+  5+  0+
* 28*  0+  0+  3+  0+  7+  0+
* 38*  1+  0+  0+  4+  0+  6+
* 48*  0+  0+  0+  9+  8+  0+
CHAPTER 9

INEQUALITIES IN THE CONSTRAINTS OF
TRANSPORTATION PROBLEM

The discussion so far is based upon the equations 2.2 through 2.5, which assume that total quantity shipped is equal to total quantity required. However, it is possible to have inequalities for the constraints. If these inequalities could be replaced by equalities then the problem could be solved as outlined earlier. Two such cases are considered below.

1. **Total availability is greater than the total demand.**

Consider the following equations:

\[ \sum_{j=1}^{m} x_{ij} \leq s_i, \quad i = 1, \ldots, m \]  \hspace{1cm} (9.1)

\[ \sum_{i=1}^{n-1} x_{ij} = d_j, \quad j = 1, \ldots, n-1 \]  \hspace{1cm} (9.2)

\[ x_{ij} \geq 0, \quad \text{all } i, j \]  \hspace{1cm} (9.3)
\[
\min Z = \sum_{i,j} c_{ij} x_{ij}
\]  

(9.4)

The first \( m \) constraints now contain a \( \leq \) sign rather than an equality sign. Physically, this simply means that more units may be available at the origins than are required at the destinations.

The inequalities can be converted to equalities by the addition of \( m \) slack variables. These slack variables may be written as \( x_{in} \), for \( i = 1, \ldots, m \).

Then the constraints become

\[
\sum_{j=1}^{n-1} x_{ij} + x_{in} = s_i, \quad i = 1, \ldots, m 
\]  

(9.5)

\[
\sum_{i=1}^{m} x_{ij} = d_j, \quad j = 1, \ldots, n-1
\]  

(9.6)

Sum (9.5) over \( i \) and subtract from the result the sum of (9.6) over \( j \).

This gives

\[
\sum_{i=1}^{m} x_{in} = \sum_{i=1}^{m} s_i - \sum_{j=1}^{n-1} d_j = d_n
\]  

(9.7)

74
Thus the total slack, i.e. the sum of the slack variables, remains constant and is the difference, denoted by $D_n$, between the origin availabilities and the destination requirements. To construct the table, simply add one more column, i.e., an additional destination for the slack. Intuitively, this approach is to be expected since the units not shipped can be considered to be shipped to origins at no cost. That is, the cost $C_{in}$ associated with the slack variable $X_{in}$ is zero.

2. **Total demand is greater than total availability.**

Consider the following problem:

\[ \sum_{j=1}^{n} x_{ij} = s_i, \quad i = 1, \ldots, m - 1 \quad (9.8) \]

\[ \sum_{i=1}^{m-1} x_{ij} \geq d_j, \quad j = 1, \ldots, n \quad (9.9) \]

\[ x_{ij} \geq 0, \quad \text{all } i, j \quad (9.10) \]

\[ \max Z = \sum_{i,j} c_{ij} x_{ij}. \quad (9.11) \]

Here, \[ \sum_{j} d_j \geq \sum_{i} s_i \]

Introduce the surplus variables $X_{mj}$, for $j = 1, \ldots, n$. Now
\[ - \sum_{j=1}^{n} x_{mj} = \sum_{j=1}^{n} D_j - \sum_{i=1}^{m-1} s_i = s_m \leq 0. \]  
(9.12)

The constraints of equation (9.8) through (9.10) can therefore be converted into the set of equations:

\[ \sum_{j=1}^{n} x_{ij} = s_i, \quad i = 1, \ldots, m - 1 \]  
(9.13)

\[ - \sum_{j=1}^{n} x_{mj} = s_m, \]  
(9.14)

\[ \sum_{i=1}^{m-1} x_{ij} - x_{mj} = D_j, \quad j = 1, \ldots, n \]  
(9.15)

The computational method is precisely the same as before except that use \( \bar{c}_{mj} = -u_m - v_j \) for computing \( \bar{c}_{mj} - c_{mj} \). This follows immediately from the dual. To solve this problem, add one more row to the table, i.e., an additional origin containing the negative of the total surplus. Here the additional cost elements \( c_{m+1,j} \) are assumed to be zero.

Thus, provided all costs are positive in the optimal solution to a problem of the form:

\[ \sum_{j=1}^{m} x_{ij} \leq s_i, \quad i = 1, \ldots, m \]  
(9.16)
\[
\sum_{i=1}^{m} x_{ij} \geq D_j, \quad j = 1, \ldots, n \quad (9.17)
\]

\[x_{ij} \geq 0, \quad \text{all } i, j \quad (9.18)\]

Max or min \( Z = \sum_{i,j} c_{ij} x_{ij} \), \( (9.19) \)

Strict equalities will hold

(a) in the destination constraints if \( Z \) is to be minimized and

(b) in the origin constraints if \( Z \) is to be maximized.

Physically, this means that if costs are minimized, no more will be shipped than necessary, and if \( Z \) is maximized, as much will be shipped as possible.
CHAPTER 10

OPERATING EXPERIENCE

The size of problem that can be solved by the method outlined in this paper is limited by the size of basis table and the amount of cost data which would fit into the fast access cores. However, the cost data may be stored on the magnetic drum or disk.

In many practical transportation situations, it is known at the start that many of the total of \( mn \) possible shipping routes are absurd. In fact, costs may not be known for many of these routes because it is certain that they would be inefficient. One would not ship to a customer in Vancouver from Montreal warehouse if there were a warehouse in Victoria, (B.C.). For problems involving a large number of cost elements, only the essential costs may be stored, all other costs not specified in the data being assumed infinite.

Virtually all applications of the transportation problem require the shipping of only whole units of the item being considered. Hence every basic feasible solution has integral values. Any optimal basic solution to transportation problem will have the property that all positive \( X_{ij} \) will be integers (12).
This integrality property is peculiar to the transportation problem. In general, an optimal solution to an arbitrary linear programming problem may not have integral values for the variables. In fact, for the variables to be integers, a linear programming problem usually becomes a nonlinear programming problem. Since an integral number of units is required at each destination, an integral number of units will be shipped.

Three different methods of selecting the element for the next iteration were tried. The results are tabulated in Table 10.1. The test data was stored in the main memory and execution times were obtained for each method for the same test data.

**METHOD 1: BEST POSITION IN COST MATRIX.**

According to usual practice with the stepping stone method, the entire cost matrix is examined at each iteration and the position is selected that gives the greatest incremental cost, \( \Delta C = U_i + V_j - C_{ij} \). A new element is placed at this position in the shipment matrix for the next iteration.

When the logical search procedure is used on a high speed computer, the time required to carry out the operations of one iteration becomes short compared to the time required to search through the entire cost data in finding the new element to be put in the basis. Thus, the total time for solution might be cut down if the time spent searching the cost data
per iteration were shortened at the expense of a larger number of iterations.

METHOD 2: FIRST WHICH WOULD REDUCE TOTAL COST.

The cost matrix is scanned row by row and the first position for which $U_i + V_j - C_{ij}$ is positive is selected. After each iteration, the search is resumed where it was broken off.

METHOD 3: BEST IN ROW OF COST MATRIX.

A complete row of the cost matrix is examined and the position in this row with the greatest incremental cost $\Delta C = U_i + V_j - C_{ij}$ is chosen. For the next iteration the next row is examined in the same way.

COMPARISON OF METHODS:

The method of searching one row at a time gives the best results. Naturally, with a different computer, a different compromise between number of iterations and time spent in searching the cost data will be optimum.

All the 3 methods were tried for a problem with 30 supplies, 260 demands and 7800 non-infinite costs on CDC 3500 computer. The results are compared with the usual rule in Table 10.i. Where,

$$\text{Ratio of timings} = \frac{\text{solution time of a method}}{\text{solution time of method 3 for 30 x 260 problem}}$$
<table>
<thead>
<tr>
<th>Method of Selecting New Element</th>
<th>Number of Iterations</th>
<th>No. of Examinations of Cost Matrix</th>
<th>Ratio of Timings*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best position in cost matrix</td>
<td>509</td>
<td>509</td>
<td>2</td>
</tr>
<tr>
<td>First which would reduce total cost</td>
<td>2200</td>
<td>29</td>
<td>2.6</td>
</tr>
<tr>
<td>Best in row of cost matrix</td>
<td>672</td>
<td>43</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 10.1: Comparison of Methods for a 30 by 260 Problem

* Obtained by dividing solution timings by the solution time of third method.
<table>
<thead>
<tr>
<th>Solution Time Ratio for</th>
<th>Problem Size $m + n$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td>Method 1/Meth 3</td>
<td>2.033</td>
</tr>
<tr>
<td>Method 2/Meth 3</td>
<td>2.624</td>
</tr>
<tr>
<td>Method 3/Meth 3</td>
<td>1.000</td>
</tr>
</tbody>
</table>

TABLE 10.2: Comparison of Solution Time Ratios
Fig. 10.3: Solution Time Ratio as a Function of Problem Size
COMPARISON OF SOLUTION TIME RATIOS.

A second set of tests were performed using a variable set of data by all the above three methods. The results are shown in Table 10.2 and Fig. 10.3. It should be noted that these results vary only in the second decimal place.
CONCLUSION

The stepping stone method described in this report was successfully implemented and tested. The results show consistently that the method presented in this report considerably reduces the solution time.


APPENDIX A

FLOW CHARTS
Note: The input/output routines are user dependent and hence the above flow chart is block diagram. However, it is coded for the computer program to enable testing.
PART 3
SEARCH FOR NEXT ELEMENT TO ENTER BASIS

250
ISWA = 0
DC = 0
IR = 0

I2 = I + 1

I2 > N
IN = 1
I2 = 1
IE = I2
ID = 1

U(IE) + V(ID) - C(IN) > DC

ISWA = 1
DC = U(IE) + V(ID) - C(IN)
J2 = ID

IN = IN + 1
ID = ID + 1

ID > N

ISWA = 0
IR = IR + 1
IR ≥ M

300
1000

94
PART 4B

355

\[ DX = \text{MINO}(X(IY), DX) \]
A(IA) = 0
F(IA) = IY
IA = IA + 1

400

N

ISWA = 0

357

Y

IH = IY

IY = K(IY)

Y

IH = K(IY)

N

IY > N

LS(IY) = 0

Y

X(IB) = DX
B(IB) = 0
G(IB) = IY
A(IA) = 1
F(IA) = IY
IA = IA + 1
IB = IB + 1

312

Y

314

96
PART 5

CHANGE VALUES OF LOOP ELEMENTS

400

ISWA = 0
ISWB = 0

IA = IA - 1

IA ≤ 0

IY = F(IA)

A(IA) = 0

N

A(IA) = 2

Y

IA = IA - 1

N

A(IA) = 1

Y

F(IA) = IY

ISWB = 0

Y

ISWB = 0

X(IY) = (IY) + DX

N

ISWB = 1

X(IY) = X(IY) - DX

500

N

ISWA = 0

Y

N

X(IY) = 0

Y

II = I(IY)
JL = J(IY)
TH1 = IY
ISWA = 1

97
PART 7B

910

Y

IY = J(IY)

LS(IY) = 0

N

IY = LS(IY) + N

IE = I(IY)

U(IE) = U(IE) - DC

Y

LS(IY) = 0

N

IH > N

N

Y

LS(IY) = IH - N

N

970

Y

IY = K(IY)

N

880

810

B(IB) = 1

G(IG) = IY

IB = IB + 1

101
Note: The input/output routines are user dependent and hence the above flow chart is block diagram. However, it is coded for the computer program to enable testing.

*** R GUPTA *** TRANSPORTATION PROBLEM ***

AUTHOR R. GUPTA

DATE WRITTEN MARCH 10, 1970

PROBLEM.

MINIMIZE COST C = SUM OVER I, J OF (C(I,J) * X(I,J))
SUBJECT TO THE CONSTRAINTS
1. SUM OVER J OF X(I,J) = S(I)
2. SUM OVER I OF X(I,J) = D(J)
3. X(I,J) >= 0

NOTE: DATA MUST HAVE SUM OVER I OF S(I) = SUM OVER J OF D(J)

PARAMETERS.

M = NUMBER OF SOURCE (PLANTS), S(I)
N = NUMBER OF DISTRIBUTION POINTS (CUSTOMERS), D(J)
D(J) = DEMAND OF COMMODITY BY CUSTOMER J
S(I) = SUPPLY OF COMMODITY AT PLANT I
I = VARIABLE FOR M (VALUE BETWEEN 1 AND M)
J = VARIABLE FOR N (VALUE BETWEEN 1 AND N)
DC = INCREMENTAL COST = U(I)+V(J)-C(I,J)

U(IE) = DUAL VARIABLE FOR ROW IE OF COST MATRIX
V(ID) = DUAL VARIABLE FOR COL ID OF COST MATRIX
K(IY) = NUMBER OF AN ELEMENT IN THE SAME ROW OF
        SHIPMENT MATRIX AS ELEMENT IY

L(IY) = NUMBER OF AN ELEMENT IN THE SAME COL OF
        SHIPMENT MATRIX AS ELEMENT IY

I(IY) = ROW OF BASIS ELEMENT
J(IY) = COL OF BASIS ELEMENT
LS(IY) = 0 IF L(IY) .LE. N
        = L(IY)-N OTHERWISE

F(I:A) = ELEMENT NUMBER OF BRANCH OR CORNER POINT
        ENCOUNTERED DURING SEARCH

A(I:A) = IDENTIFIES F(I:A) ENCOUNTERED DURING SEARCH AS
         ** MAX 2*M+N-2
         A(I:A)=0 = CORNER POINT
         =1 = BRANCH POINT
         =2 = BRANCH POINT FROM WHICH A NEW
         PATH WAS SEARCHED

G(I:B) = ELEMENT NUMBER OF BRANCH POINT ENCOUNTERED DURING ** MAX M-1
ROUTINE GPXPOSE

C SEARCH
C
C B(IB) - INDICATES WHETHER BRANCH POINT WAS FOUND DURING **MAX M-1
C ROW SEARCH = B(IB)=0 OR
C COL SEARCH = B(IB)=1.
C
C X(IB) - VALUE OF SMALLEST ALTERNATE CORNER OR BRANCH **MAX M-1
C ELEMENT OF THE PATH FROM THE STARTING POINT
C OF A TREE TO AND INCLUDING ELEMENT G(IB).
C
C C(IN) - UNIT COST FROM PLANT I TO CUSTOMER J **MAX M*N
C IN = INDEX FOR COST, C(IN) = J+N*(I-1)
C
C E(IZ) - DENOTES WHETHER AN ENTRY IN THE BASIS TABLE HAS **MAX N
C BEEN MADE (E(IZ)=1) OR
C HAS NOT BEEN MADE (E(IZ)=0)
C
C FOR COL IZ OF SHIPPING MATRIX
C (USED ONLY IN FORMING INITIAL BASIS).
C
C C0 - COST FOR CURRENT BASIS ELEMENT
C
C U0 - VALUE OF U FOR ROW OF CURRENT ELEMENT
C V0 - VALUE OF U FOR COL OF CURRENT ELEMENT
C
C IH1 - BASIS TABLE ENTRY NUMBER OF ELEMENT DROPPED FROM
C BASIS IN A PARTICULAR ITERATION.
C
C I1 - ROW ENTRY NUMBER OF ELEMENT DROPPED FROM
C BASIS IN A PARTICULAR ITERATION
C
C J1 - COL ENTRY NUMBER OF ELEMENT DROPPED FROM
C BASIS IN A PARTICULAR ITERATION.
C
C IP - REGISTERS FOR TEMPORARY STORAGE AND COUNTING
C
C IQ - REGISTERS FOR TEMPORARY STORAGE AND COUNTING
C IR - REGISTERS FOR TEMPORARY STORAGE AND COUNTING
C IS - REGISTERS FOR TEMPORARY STORAGE AND COUNTING
C IT - REGISTERS FOR TEMPORARY STORAGE AND COUNTING
C IU - REGISTERS FOR TEMPORARY STORAGE AND COUNTING
C IV - REGISTERS FOR TEMPORARY STORAGE AND COUNTING
C
C IH2 - BASIS TABLE ENTRY NUMBER OF ELEMENT ADDED TO BASIS
C IN THIS ITERATION.
C
C I2 - ROW ENTRY NUMBER OF ELEMENT ADDED TO BASIS
C IN THIS ITERATION
C
C J2 - COL ENTRY NUMBER OF ELEMENT ADDED TO BASIS
C IN THIS ITERATION
C
C IH - BASIS TABLE ENTRY NUMBER OF FIRST ELEMENT EXAMINED
C IN A PARTICULAR ROW OR COL OF THE SHIPMENT MATRIX
C DURING LOGICAL SEARCH.
C
C IPALL = 3 PRINT EVERYTHING
C = 4 PRINT FINAL ITERATION ONLY
C
C KAUNT = NUMBER OF ITERATION
C
C --------------------------------------------------------------
C
C COMMON IPALL,KOST,KSUMS
C
C INTEGER C(1) ***** SIZE M*N *****
C
C INTEGER S(1),U(1) ***** SIZE M *****
ROUTINE GXPOSE

C

INTEGER D(1),V(1),E(1)

C

INTEGER I(1),J(1),K(1),LS(1),X(1)

C

INTEGER G(1),B(1)

C

INTEGER A(1),F(1)

C

INTEGER DC,DX,C0,U0,V0

C

KR = 60
KW = 61
KSUMS = 0
KSUMD = 0

C

===============================================

C

* PART 1 * INPUT SECTION — READ IN THE DATA *

C

===============================================

C

DO 20 IK=1,M
F(IK)=0
A(IK)=0

B(IK)=0
G(IK)=0
KSUMS = KSUMS + S(IK)
20 CONTINUE

C

===============================================

C

* PART 2 * FORM INITIAL BASIC FEASIBLE SOLUTION **

C

===============================================

C

IP = N
IV = M-1
MN1 = M+N-1
IW = MN1
MN2 = M+2+N-2
M1 = 2*N

U0 = 0
INF = 2**20
IE = 1
KAUNT = 0

C

W(IE)= 0
C
DO 100 IZ=1,N
KSUMD = KSUMD + D(IZ)
100 E(IZ) = 0

C
IF (KSUMS-KSUMD) .GE. 103,105,103
103 WRITE (104) KSUMS,KSUMD
104 FORMAT (1H10,10HSUM OF S = ,I6,3X,10HSUM OF D = ,I6/)
GO TO 2000

C
105 CONTINUE
ISWA = 0

C
115 IN = (IE-1)*N+1
IZ = 1
C0 = INF
120 IF(C(IN)-C0) 125,125,135
125 IF(E(IZ)) 135,138,135
130 ID = IZ
C0 = C(IN)
135 IF(IZ-N) 140,145,140
140 IN = IN+1
IZ = IZ+1
GO TO 120

C
145 IZ = ID
E(IZ)= 1
V(ID)= C0-U0
V0 = V(ID)

C
IF(ISWA) 155,150,155

150 ISKA = 1
IR = ID
IS = ID
IT = 0
IQ = ID
GO TO 160

C
155 IS = IR
IT = 0
IR = ID

C
160 IF(IY) 165,175,165
165 IF(D(ID)-S(IE)) 170,170,175
170 IY = ID
GO TO 220

C
175 IY = IQ
K(IY)= IR

C
IQ = ID
IY = ID

C
180 I(IY)= IE
K(IY)= K0

72/08/
3ROUTINE GPOSE

J(IY)= ID
LS(IY)=IT
X(IY)= S(IE)
D(ID)=D(ID)-S(IE)
S(IE)= i
IW = IW+1

C

IF(IW) 185, 225, 185
185 IE = IE+1
IN = (IE-1)+N+ID
C0 = C(IN)
U(IE) = U-C0

U0 = U(IE)
IP = IP+1
IF(IR-N) 195, 195, 190
190 IT = IR-N
GO TO 200

C

195 IT = 0
200 IR = IP
IS=IR
IF(I-E=N) 205, 215, 205
205 IF(S(IE)-D(ID)) 210, 210, 215
210 IY = IP
GO TO 180

C

215 IY = IQ
LS(IY)= IP-N

C

IQ = IP
IY = IP

C

220 I(IY) = IE
J(IY) = ID
K(IY) = IS
LS(IY)= IT
X(IY) = D(ID)
S(IE) = S(IE)-D(ID)
D(ID) = O
IW = IW-1

C

IF(IW) 230, 225, 230
225 IN = 1
I2 = Y
GO TO 250

C

230 IV = IV-1
GO TO 115

C

250 CONTINUE

C

KAUNT = KAUNT+1
IGO = 1
IF(IPALL=3) 8253, 1002, 8253

IR GEORGE WILLIAMS UNIVERSITY
Routine GXPOSE

8253 CONTINUE

C
C * PART 3 * SEARCH FOR NEXT ELEMENT TO ENTER BASIS
C
C
C
ISWA = 0
DC = 0
IR = 0
255 I2 = I2+1
   IF(I2=M) 265,265,260
260 IN = 1
   I2 = 1
265 IE = I2
   ID = 1
270 IDC = U(IIE)+V(ID)-C(IN)
   IF(IDC-DC) 280,280,275
275 ISWA = 1
   DC = IDC
   J2 = ID
280 CONTINUE

C
   IN = IN+1
   ID = ID+1
   IF(ID=M) 270,270,285
285 IF(ISWA) 300,295,300
295 IR = IR+1
   IF(IR=M) 255,1000,1000

C
300 CONTINUE
   DO 301 IJ=1,NN1
C
301 E(IJ) = X(IJ)

C
C * PART 4 * FIND BASIC LOOP INCLUDING NEW ELEMENTS *
C
C
C
IA = 1
   IB = 1
   IY = J2
   IH = J2
   DX = INF
   ISWA = 0

C
305 IF(LS(IY)) 315,310,315
310 IF(IY-K(IY)) 355,312,355
311 CONTINUE

C
   WRITE (KW,8311)
8311 FORMAT (IH,15H* PART 4 * IB=0 /)

IR: GEORGE WILLIAMS UNIVERSITY
OUTLINE GXPOSE

GO TO 2000

C
312 IB = IB+1
C
IF (IB) 311,311,313
313 IY = G(IB)
C
DX = E(IB)
A(IA) = 2
F(IA) = IY
IA = IA+1
IF(B(IB)+1) 314,357,314
314 IH = IY
C
GO TO 325
C
315 CONTINUE
IF(IY=K(IY)) 320,325,320
C
320 CONTINUE
C
E(IB) = MIN0(X(IY),DX)
B(IB) = 1
G(IB) = IY
A(IA) = 1
F(IA) = IY
IA = IA+1
IB = IB+1
325 IF(LS(IY)) 335,330,335
330 IY = J(IY)
GO TO 340
C
335 IY = LS(IY)+N
340 IF(I(IY)=12) 345,342,345
342 ISWA = 1
GO TO 355
C
345 CONTINUE
IF(IH=N) 305,305,350
350 IF(LS(IY)-(IH=N)) 315,310,315
355 DX = MIN0(X(IY),DX)
C
A(IA) = 0
F(IA) = IY
IA = IA+1
IF(ISWA) 400,357,400
C
357 IH = IY
360 IY = K(IY)
IF(IH=K(IY)) 380,365,380
365 IF(IY=N) 375,378,375
379 IF(LS(IY)) 375,312,375
375 A(IA) = 0
F(IA) = IY
IA = IA+1
OUTINE GXPOSE

GO TO 314

C
380 IF(IY-N) 385,385,390
385 IF(LS(IY)) 390,360,390
C
390 CONTINUE
C
E(IB)=DX
B(IB)= 0
G(IB)= IY
A(IA)= 1
F(IA)= IY
IA = IA+1
IB = IB+1
GO TO 360

C
400 CONTINUE
C
C
* PART 5 * CHANGE VALUES OF LOOP ELEMENTS *
C
C
ISWA = 0
ISWB = 0
C
405 IA = IA-1
410 IF(I) 500,500,410
410 IY = F(I)
420 IF(A(IA)) 420,450,420
420 IF(A(IA)-2) 420,430,420
430 IA = IA-1
430 IF(I) 500,500,430
430 IF(A(IA)-1) 430,440,430
440 IF(F(IA)-IY) 430,450,430
450 IF(ISWB) 460,470,460

460 ISWB = 0
460 X(IY)= X(IY)+DX
GO TO 405
C
470 ISWB = 1
470 X(IY)= X(IY)-DX
480 IF(ISWA) 495,480,405
480 IF(X(IY)) 435,490,405
C
490 I1 = I(IY)
490 J1 = J(IY)
490 IH1 = IY
490 ISWA = 1
GO TO 435
C
500 CONTINUE
C
C
R GEORGE WILLIAMS UNIVERSITY
C * PART 6 * * MODIFY BASIS TABLE *

C =================================================================================

C      IM2 = IH1
C      IV = IH1
C      IC = IH1

C      IF(J1-J2) 550,510,550
C      510 ISWA = 0
C      520 KP = IH1
C      530 IF(K(KP)-IH1) 540,710,540
C      540 KP = K(KP)
C      GO TO 530
C

C      IF(IH1-N) 560,560,570
C      560 ISWA = 1
C      GO TO 520
C

C      570 KP = IH1
C      580 IF(LS(KP)) 605,590,605
C      590 KP = J1
C      GO TO 580
C

C      600 KP = LS(KP)+N
C      GO TO 580
C

C      605 IF(LS(KP)-(IH1-N)) 600,610,600
C      610 LS(KP)= LS(IY)
C      ISWB = N

C      620 KP = J2
C      LS(IC)= LS(KP)
C      J(IC) = J2
C      LS(KP)= IC-N
C      IF(ISWB) 640,630,640

C      630 IF(I1-I2) 510,790,510
C      640 IF(I(IC)-I2) 650,690,650
C      650 IF(I1-I2) 720,660,720
C      660 IF(IP-IY) 670,700,670
C      670 KP = IP
C      680 I(IC) = I2
C      K(IC) = K(KP)
C      K(KP) = IC
C      GO TO 790
C

C      690 KP = IH1
C      GO TO 680
C

C      700 KP = IH2
C      GO TO 680
C

C      710 K(KP) = K(IY)
C      IF(ISWA) 750,720,750

R GEORGE WILLIAMS UNIVERSITY
OUTLINE GXXPOSE

720 KP = 1
730 IF(I(KP)-I2) 740, 680, 740
740 KP = KP+1
GO TO 730

C
750 IC = LS(IY)+N
IH2 = IC
KP = IH2
760 IF(K(KP)-IH2) 770, 780, 770
770 KP = K(KP)
GO TO 760

C
780 K(KP) = IY
K(IY) = K(IC)
I(IY) = I(IC)
IP = K(IY)
LS(IY) = LS(IC)
X(IY) = X(IC)
ISW0 = 1
GO TO 620

C
790 X(IC) = DX
GO TO 800

C
C
C ================
C * PART 7 * COMPUTE DUAL VARIABLES FOR NEW BASIS*
C
C ================

800 CONTINUE
IB = 1
IY = IH2
IE = I2

C
U(IE) = U(IE)-DC
IF(K(IY)-IY) 810, 250, 810

C
810 IH = IY
820 IY = K(IY)
ID = J(IY)
V(ID) = V(ID)+DC
IF(K(IY)-IH) 830, 660, 830

830 IF(IY-N) 840, 640, 650
840 IF(LS(IY)) 850, 620, 650
850 B(IB) = 1
G(IB) = IY
IB = IB+1
GO TO 820

C
860 IF(IY-N) 870, 670, 900
870 IF(LS(IY)) 900, 680, 900
880 ID = IB-1
IF(IB) 890, 250, 890
890 IY = G(IB)

R GEORGE WILLIAMS UNIVERSITY
OUTLINE GXPOSE

IF(R(IY)-1) 970,810,900
900 IH = IY
910 IF(LS(IY)) 920,930,920
920 IY = LS(IY)+N
GO TO 940

C

930 IY = J(IY)
940 IE = I(IY)
U(IE) = U(IE)-DC
IF(IN==N) 950,950,960
950 IF(LS(IY)) 980,970,980
960 IF(LS(IY)-(IH-N)) 980,970,980
970 IF(IY-K(IY)) 810,880,810
980 IF(IY-K(IY)) 990,910,990
990 B(IB) = 1
6(IB) = IY
IB = IB+1
GO TO 910

C

1000 CONTINUE
100 CONTINUE

GO = 4

C

==============================================
C * PART B *  ** PRINT FINAL SOLUTION **
C

==============================================

1002 CONTINUE

C

***** CALCULATE S AND D *****
C
DO 1003 IJ=1,N
1003 D(IJ) = 0

C

DO 1004 IJ=1,M

1004 S(IJ) = 0

C

DO 1060 KT=1,MN1
IJ=J(KT)
D(IJ)=D(IJ)+X(KT)
IJ=I(IJ)
S(IJ)=S(IJ)+X(KT)
1060 CONTINUE

C

KSUMS = 0
KSUMD = 0

C

DO 1075 IJ=1,N
1075 KSUMD = KSUMD+D(IJ)
C

DO 1080 IJ=1,M
1080 KSUMS = KSUMS + S(IJ)
C

R GEORGE WILLIAMS UNIVERSITY
ROUTINE EXPOSE

C ** TOTAL S = TOTAL D IF NOT ERROR CANCEL JOB **

C IF (KSUMS=KSUMD) 103,1090,103
1090 CONTINUE

C IF (IPALL=3) 2000,1093,1092
1092 IF(IPALL=5) 1093,2000,2000
1093 CONTINUE

C WRITE (KW,1095) KAUNIT,M,N
1095 FORMAT(1H1,14H*** ITERATION, ,I4,5H *** ,
1 22H NUMBER OF ROWS =M =,I4,22H NUMBER OF COLS -N =I4/)

C ***** BASIS TABLE *****

C WRITE (KW,1100)
1100 FORMAT (12X, 22H**** BASIS TABLE **** )

C WRITE (KW,1110)

C WRITE (KW,1120)
1120 FORMAT ( 83H ***KT*,...,I(KT)*,...,J(KT)*,...,KS(KT)*
1 A(*,X(KT)),*A(*,F(KT)),*A(*,A(KT))

C WRITE (KW,1110)

C DO 1130 KT=1,4N1
1125 FORMAT (1H, 12I10)
1130 CONTINUE

C WRITE (KW,1110)

C WRITE(KW,1131) (U(IP),IP=1,M)
1131 FORMAT (6H **U**,2515)
WRITE(KW,1132) (V(IP),IP=1,N)
1132 FORMAT (6H **V**,2515)
WRITE(KW,1134) (F(IP),IP=1,N)
1134 FORMAT (6H **F**,2515)
WRITE(KW,1135) (G(IP),IP=1,M1)
1135 FORMAT (6H **G**,2515)

C WRITE (KW,1160) I2,J2,I1,J1,1H1
1160 FORMAT (/21H ELEMENTS ADDED I2 =, I4,6H J2 =, I4/
1 12H ELEMENTS Dropped I1 =,I4,5H J1 =,I4,7H IH1 = ,I4)

C ***** SHIPMENT MATRIX *****

C KSUMS = 0
WRITE (KW,1180)
1180 FORMAT(/21X,25H**** SHIPMENT MATRIX ****)
WRITE (KW,1110)

C DO 1300 IROW=1,M

R GEORGE WILLIAMS UNIVERSITY
ROUTINE GEXPOSE

C

DO 1220 KZ=1,N
1220 E(KZ) = 0

C

DO 1240 KZ=1,MN1
1240 CONTINUE

C

WRITE(KW,1250) S(IROW), ( E(IJ), IJ=1,N)
1250 FORMAT(1H,1H*,15,1H*,24(1H*,I4))

C

WRITE(KW,1110)

C

WRITE(KW,1250) (D(IJ), IJ=1,N)
WRITE(KW,1110)

C

***** COST MATRIX *****

KOST = 0
WRITE(KW,1420)

1420 FORMAT(/12X,22H***** COST MATRIX *****)
WRITE(KW,1110)

C

DO 1500 IRW=1,M

DO 1430 KZ=1,N
1430 E(KZ) = 0

C

DO 1470 KZ=1,MN1
1470 CONTINUE

C

WRITE(KW,1250) U(IRW), (E(IJ), IJ=1,N)
1500 CONTINUE

C

WRITE(KW,1110)

C

WRITE(KW,1250) (V(IJ), IJ=1,N)
WRITE(KW,1110)

C

1510 FORMAT (11H TOTAL COST, I8/)

C

***** ELEMENT MATRIX *****

C

WRITE(KW,1620)

1620 FORMAT(/12X,24H***** ELEMENT MATRIX *****)
WRITE(KW,1110)

C

DO 1700 IRW = 1,M

DO 1630 KZ = 1,N
1630 E(KZ) = 0

R GEORGE WILLIAMS UNIVERSITY
ROUTINE GXPOSE

C
DO 1670 KZ = 1, MN1
   IF (IRON - I(KZ)) 1670, 1650, 1670
1650 NCOL = J(KZ)
   E(NCOL) = KZ
1670 CONTINUE
C
   WRITE (KW, 1250) IROW, (E(IJ), IJ = 1, N)
1700 CONTINUE
C
   WRITE (KW, 1110)
C
C
   IF (IGO - 1) 2000, 8253, 2000
2000 CONTINUE
C
   RETURN
END
PROGRAM GUPTA (INPUT, OUTPUT, TAPF40=INPUT, TAPE61=OUTPUT).

COMMON IPALL, KOST, KSUMS
INTEGER C(900), S(30), U(30), D(30), V(30), E(59), F(88), A(88)
INTEGER I(59), J(59), K(59), LS(59), X(59), G(29), R(29)

KR = 60
KW = 61
IBLANK = 0
IPALL = 3

**** READ SIZE OF MATRIX ****

DO 2000 I=1,5
READ(KR, 10) M, N
10 FORMAT (16I5)
WRITE(KW, 15) M, N
MNI = M+N-1
IP1 = 1

**** READ COST MATRIX ****

DO 20 IK=1, M
IP2 = IP1+N-1
READ(KR, 10) (C(IP), IP=IP1, IP2)
WRITE(KW, 16) (C(IP), IP=IP1, IP2)
15 FORMAT (22H1NUMFR OF ROWS -M =, I4/22H NUMBER OF COLS -N =I4/)
16 FORMAT (1H, 12I10)
20 IP1 = IP1+N
WRITE(KW, 15) M, N

**** READ SUPPLY AND DEMAND ****

READ (KR, 10) (D(IP), IP=1, M)
WRITE (KW, 32) (D(IP), IP=1, M)
READ (KR, 10) (S(IP), IP=1, M)
WRITE (KW, 33) (S(IP), IP=1, M)

32 FORMAT (6H **D**, 2515)
33 FORMAT (6H **S**, 2515)

**** GO TO TRANSPORTATION SUBROUTINE ****


2000 CONTINUE

STOP
END